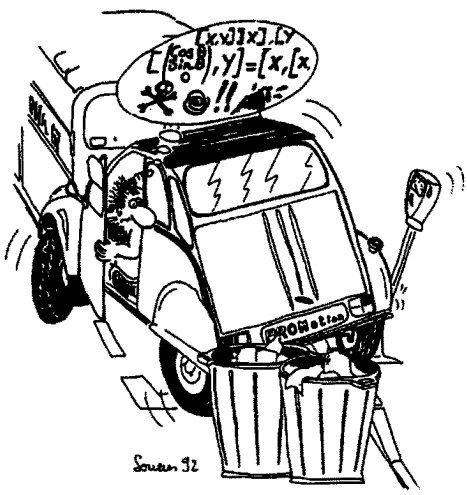
Lecture Notes
in Control and Information Sciences 229

Editor: M. Thoma

J.-P. Laumond (Ed.)

Robot Motion Planning and Control



Springer

Editor

Dr J.-P. Laumond
Centre National de la Recherche Scientifique
Laboratoire d'Analyse et d'Architecture des Systemes
7, avenue du Colonel Roche
31077 Toulouse Cedex
FRANCE

ISBN 3-540-76219-1 Springer-Verlag Berlin Heidelberg New York

British Library Cataloguing in Publication Data
Robot motion planning and control. - (Lecture notes in
control and information sciences ; 229)
1.Robots - Motion 2.Robots - Control systems
I.Laumond, J.-P.
629.8'92
ISBN 35400762191

Library of Congress Cataloging-in-Publication Data
Robot motion planning and control. / J. -P. Laumond (ed.).
p. cm. - - (Lecture notes in control and information sciences ; 229)
Includes bibliographical references.
ISBN 3-540-76219-1 (pbk. : alk. paper)
1. Robots- -Motion. 2. Robots- -Control systems. I. Laumond, J. -P. (Jean-Paul) II. Series
TJ211.4.R63 1998 97-40560
629.8'92- -dc21 CIP

Typesetting: Camera ready by editor
Printed and bound at the Athenæum Press Ltd, Gateshead
69/3830-543210 Printed on acid-free paper

Foreword

How can a robot decide what motions to perform in order to achieve tasks in the physical world ?

The existing industrial robot programming systems still have very limited motion planning capabilities. Moreover the field of robotics is growing: space exploration, undersea work, intervention in hazardous environments, servicing robotics ... Motion planning appears as one of the components for the necessary autonomy of the robots in such real contexts. It is also a fundamental issue in robot simulation software to help work cell designers to determine collision free paths for robots performing specific tasks.

Robot Motion Planning and Control requires interdisciplinarity

The research in robot motion planning can be traced back to the late 60's, during the early stages of the development of computer-controlled robots. Nevertheless, most of the effort is more recent and has been conducted during the 80's (*Robot Motion Planning*, J.C. Latombe's book constitutes the reference in the domain).

The position (configuration) of a robot is normally described by a number of variables. For mobile robots these typically are the position and orientation of the robot (i.e. 3 variables in the plane). For articulated robots (robot arms) these variables are the positions of the different joints of the robot arm. A motion for a robot can, hence, be considered as a path in the configuration space. Such a path should remain in the subspace of configurations in which there is no collision between the robot and the obstacles, the so-called free space. The motion planning problem asks for determining such a path through the free space in an efficient way.

Motion planning can be split into two classes. When all degrees of freedom can be changed independently (like in a fully actuated arm) we talk about *holonomic motion planning*. In this case, the existence of a collision-free path is characterized by the existence of a connected component in the free configuration space. In this context, motion planning consists in building the free configuration space, and in finding a path in its connected components.

Within the 80's, Roboticians addressed the problem by devising a variety of heuristics and approximate methods. Such methods decompose the configuration space into simple cells lying inside, partially inside or outside the free space. A collision-free path is then searched by exploring the adjacency graph of free cells.

In the early 80's, pioneering works showed how to describe the free configuration space by algebraic equalities and inequalities with integer coefficients (i.e. as being a semi-algebraic set). Due to the properties of the semi-algebraic sets induced by the Tarski-Seidenberg Theorem, the connectivity of the free configuration space can be described in a combinatorial way. From there, the road towards methods based on Real Algebraic Geometry was open. At the same time, Computational Geometry has been concerned with combinatorial bounds and complexity issues. It provided various exact and efficient methods for specific robot systems, taking into account practical constraints (like environment changes).

More recently, with the 90's, a new instance of the motion planning problem has been considered: planning motions in the presence of kinematic constraints (and always amidst obstacles). When the degrees of freedom of a robot system are not independent (like e.g. a car that cannot rotate around its axis without also changing its position) we talk about *nonholonomic motion planning*. In this case, any path in the free configuration space does not necessarily correspond to a feasible one. Nonholonomic motion planning turns out to be much more difficult than holonomic motion planning. This is a fundamental issue for most types of mobile robots. This issue attracted the interest of an increasing number of research groups. The first results have pointed out the necessity of introducing a Differential Geometric Control Theory framework in nonholonomic motion planning.

On the other hand, at the motion execution level, nonholonomy raises another difficulty: the existence of stabilizing smooth feedback is no more guaranteed for nonholonomic systems. Tracking of a given reference trajectory computed at the planning level and reaching a goal with accuracy require nonstandard feedback techniques.

Four main disciplines are then involved in motion planning and control. However they have been developed along quite different directions with only little interaction. The coherence and the originality that make motion planning and control a so exciting research area come from its *interdisciplinarity*. It is necessary to take advantage from a common knowledge of the different theoretical issues in order to extend the state of the art in the domain.

About the book

The purpose of this book is not to present a current state of the art in motion planning and control. We have chosen to emphasize on recent issues which have been developed within the 90's. In this sense, it completes Latombe's book published in 1991. Moreover an objective of this book is to illustrate the necessary interdisciplinarity of the domain: the authors come from Robotics,

Computational Geometry, Control Theory and Mathematics. All of them share a common understanding of the robotic problem.

The chapters cover recent and fruitful results in motion planning and control. Four of them deal with nonholonomic systems; another one is dedicated to probabilistic algorithms; the last one addresses collision detection, a critical operation in algorithmic motion planning.

Nonholonomic Systems The research devoted to nonholonomic systems is motivated mainly by mobile robotics. The first chapter of the book is dedicated to nonholonomic path planning. It shows how to combine geometric algorithms and control techniques to account for the nonholonomic constraints of most mobile robots. The second chapter develops the mathematical machinery necessary to the understanding of the nonholonomic system geometry; it puts emphasis on the nonholonomic metrics and their interest in evaluating the combinatorial complexity of nonholonomic motion planning. In the third chapter, optimal control techniques are applied to compute the optimal paths for car-like robots; it shows that a clever combination of the maximum principle and a geometric viewpoint has permitted to solve a very difficult problem. The fourth chapter highlights the interactions between feedback control and motion planning primitives; it presents innovative types of feedback controllers facing nonholonomy.

Probabilistic Approaches While complete and deterministic algorithms for motion planning are very time-consuming as the dimension of the configuration space increases, it is now possible to address complicated problems in high dimension thanks to alternative methods that relax the completeness constraint for the benefit of practical efficiency and probabilistic completeness. The fifth chapter of the book is devoted to probabilistic algorithms.

Collision Detection Collision checkers constitute the main bottleneck to conceive efficient motion planners. Static interference detection and collision detection can be viewed as instances of the same problem, where objects are tested for interference at a particular position, and along a trajectory. Chapter six presents recent algorithms benefiting from this unified viewpoint.

The chapters are self-contained. Nevertheless, many results just mentioned in some given chapter may be developed in another one. This choice leads to repetitions but facilitates the reading according to the interest or the background of the reader.

On the origin of the book

All the authors of the book have been involved in PROMotion. PROMotion was a European Project dedicated to robot motion planning and control. It has progressed from September 1992 to August 1995 in the framework of the Basic Research Action of ESPRIT 3, a program of research and development in Information Technologies supported by the European Commission (DG III). The work undertaken under the project has been aimed at solving concrete problems. Theoretical studies have been mainly motivated by a practical efficiency. Research in PROMotion has then provided methods and their prototype implementations which have the potential of becoming key components of recent programs in advanced robotics.

In few numbers, PROMotion is a project whose cost has been 1.9 MEcus[1] (1.1 MEcus supported by European Community), for a total effort of more than 70 men-year, 179 research reports (most them have been published in international conferences and journals), 10 experiments on real robot platforms, an International Spring school and 3 International Workshops. This project has been managed by LAAS–CNRS in Toulouse; it has involved the "Universitat Politecnica de Catalunya" in Barcelona, the "Ecole Normale Supérieure" in Paris, the University "La Sapienza" in Roma, the Institute INRIA in Sophia-Antipolis and the University of Utrecht.

J.D. Boissonnat (INRIA, Sophia-Antipolis), A. De Luca (University "La Sapienza" of Roma), M. Overmars (Utrecht University), J.J. Risler (Ecole Normale Supérieure and Paris 6 University), C. Torras (Universitat Politecnica de Catalunya, Barcelona) and the author make up the steering committee of PROMotion. This book benefits from contributions of all these members and their co-authors and of the work of many people involved in the project.

On behalf of the project committee, I thank J. Wejchert (Project officier of PROMotion for the European Community), A. Blake (Oxford University), H. Chochon (Alcatel) and F. Wahl (Braunschweig University) who acted as reviewers of the project during three years. Finally I thank J. Som for her efficient help in managing the project and M. Herrb for his help in editing this book.

Jean-Paul Laumond
LAAS-CNRS, Toulouse
August 1997

[1] US $ 1 $\approx$ 1 Ecu

List of Contributors

A. Bellaïche
Département de Mathématiques
Université de Paris 7
2 Place Jussieu
75251 Paris Cedex 5
France
abellaic@mathp7.jussieu.fr

J.D. Boissonnat
INRIA Centre de Sophia Antipolis
2004, Route des Lucioles BP 93
06902 Sophia Antipolis Cedex,
France
boissonn@sophia.inria.fr

A. De Luca
Dipartimento di Informatica
e Sistemistica
Universitá di Roma "La Sapienza"
Via Eudossiana 18
00184 Roma
Italy
adeluca@giannutri.caspur.it

F. Jean
Institut de Mathématiques
Université Pierre et Marie Curie
Tour 46, 5ème étage, Boite 247
4 Place Jussieu
75252 Paris Cedex 5
France
jean@math.jussieu.fr

P. Jiménez
Institut de Robòtica
i Informàtica Industrial
Gran Capità, 2
08034-Barcelona
Spain
jimenez@iri.upc.es

F. Lamiraux
LAAS-CNRS
7 Avenue du Colonel Roche
31077 Toulouse Cedex 4
France
lamiraux@laas.fr

J.P. Laumond
LAAS-CNRS
7 Avenue du Colonel Roche
31077 Toulouse Cedex 4
France
jpl@laas.fr

G. Oriolo
Dipartimento di Informatica
e Sistemistica
Universitá di Roma "La Sapienza"
Via Eudossiana 18
00184 Roma
Italy
oriolo@giannutri.caspur.it

M. H. Overmars
Department of Computer Science,
Utrecht University
P.O.Box 80.089,
3508 TB Utrecht,
the Netherlands
markov@cs.ruu.nl

J.J. Risler
Institut de Mathématiques
Université Pierre et Marie Curie
Tour 46, 5ème etage, Boite 247
4 Place Jussieu
75252 Paris Cedex 5
France
risler@math.jussieu.fr

C. Samson
INRIA Centre de Sophia Antipolis
2004, Route des Lucioles BP 93
06902 Sophia Antipolis Cedex,
France
Claude.Samson@sophia.inria.fr

S. Sekhavat
LAAS-CNRS
7 Avenue du Colonel Roche
31077 Toulouse Cedex 4
France
sepanta@laas.fr

P. Souères
LAAS-CNRS
7 Avenue du Colonel Roche
31077 Toulouse Cedex 4
France
soueres@laas.fr

P. Švestka
Department of Computer Science,
Utrecht University
P.O.Box 80.089,
3508 TB Utrecht,
the Netherlands
petr@cs.ruu.nl

F. Thomas
Institut de Robòtica
i Informàtica Industrial
Gran Capità, 2
08034-Barcelona
Spain
thomas@iri.upc.es

C. Torras
Institut de Robòtica
i Informàtica Industrial
Gran Capità, 2
08034-Barcelona
Spain
torras@iri.upc.es

Table of Contents

Guidelines in Nonholonomic Motion Planning for Mobile Robots

J.P. Laumond, S. Sekhavat and F. Lamiraux

LAAS-CNRS, Toulouse

1 Introduction

Mobile robots did not wait to know that they were nonholonomic to plan and execute their motions autonomously. It is interesting to notice that the first navigation systems have been published in the very first International Joint Conferences on Artificial Intelligence from the end of the 60's. These systems were based on seminal ideas which have been very fruitful in the development of robot motion planning: as examples, in 1969, the mobile robot Shakey used a grid-based approach to model and explore its environment [61]; in 1977 Jason used a visibility graph built from the corners of the obstacles [88]; in 1979 Hilare decomposed its environment into collision-free convex cells [30].

At the end of the 70's the studies of robot manipulators popularized the notion of configuration space of a mechanical system [53]; in this space the "piano" becomes a point. The motion planning for a mechanical system is reduced to path finding for a point in the configuration space. The way was open to extend the seminal ideas and to develop new and well-grounded algorithms (see Latombe's book [42]).

One more decade, and the notion of nonholonomy (also borrowed from Mechanics) appears in the literature [44] on robot motion planning through the problem of car parking which was not solved by the pioneering mobile robot navigation systems. Nonholonomic Motion Planning then becomes an attractive research field [52].

This chapter gives an account of the recent developments of the research in this area by focusing on its application to mobile robots.

Nonholonomic systems are characterized by constraint equations involving the time derivatives of the system configuration variables. These equations are non integrable; they typically arise when the system has less controls than configuration variables. For instance a car-like robot has two controls (linear and angular velocities) while it moves in a 3-dimensional configuration space. As a consequence, any path in the configuration space does not necessarily correspond to a feasible path for the system. This is basically why the purely geometric techniques developed in motion planning for holonomic systems do not apply directly to nonholonomic ones.

While the constraints due to the obstacles are expressed directly in the manifold of configurations, nonholonomic constraints deal with the tangent space. In the presence of a link between the robot parameters and their derivatives, the first question to be addressed is: does such a link reduce the *accessible* configuration space ? This question may be answered by studying the structure of the distribution spanned by the Lie algebra of the system controls.

Now, even in the absence of obstacle, planning nonholonomic motions is not an easy task. Today there is no general algorithm to plan motions for any nonholonomic system so that the system is guaranteed to exactly reach a given goal. The only existing results are for approximate methods (which guarantee only that the system reaches a neighborhood of the goal) or exact methods for special classes of systems; fortunately, these classes cover almost all the existing mobile robots.

Obstacle avoidance adds a second level of difficulty. At this level we should take into account both the constraints due to the obstacles (i.e., dealing with the configuration parameters of the system) and the nonholonomic constraints linking the parameter derivatives. It appears necessary to combine geometric techniques addressing the obstacle avoidance together with control theory techniques addressing the special structure of the nonholonomic motions. Such a combination is possible through topological arguments.

The chapter may be considered as self-contained; nevertheless, the basic necessary concepts in differential geometric control theory are more developed in Bellaïche–Jean–Risler's chapter.

Finally, notice that Nonholonomic Motion Planning may be consider as the problem of planning *open loop* controls; the problem of the feedback control is the purpose of DeLuca–Oriolo–Samson's chapter.

2 Controllabilities of mobile robots

The goal of this section is to state precisely what kind of controllability and what level of mobile robot modeling are concerned by motion planning.

2.1 Controllabilities

Let us consider a n-dimensional manifold, $\mathcal{U}$ a class of functions of time t taking their values in some compact sub-domain $\mathcal{K}$ of $\mathbf{R}^m$. The control systems Σ considered in this chapter are differential systems such that

$$\dot{X} = f(X)u + g(X).$$

u is the control of the system. The $i-$th column of the matrix $f(X)$ is a vector field denoted by f_i. $g(X)$ is called the drift. An *admissible* trajectory is a

solution of the differential system with given initial and final conditions and u belonging to $\mathcal{U}$.

The following definitions use Sussmann's terminology [83].

Definition 1. *Σ is locally controllable from X if the set of points reachable from X by an admissible trajectory contains a neighborhood of X. It is small-time controllable from X if the set of points reachable from X before a given time T contains a neighborhood of X for any T.*

A control system will be said to be small-time controllable if it is small-time controllable from everywhere.

Small-time controllability clearly implies local controllability. The converse is false.

Checking the controllability properties of a system requires the analysis of the control Lie algebra associated with the system. Considering two vector fields f and g, the Lie bracket $[f, g]$ is defined as being the vector field $\partial f.g - \partial g.f$ [1]. The following theorem (see [82]) gives a powerful result for symmetric systems (i.e., $\mathcal{K}$ is symmetric with respect to the origin) without drift (i.e, $g(X) = 0$).

Theorem 2.1. *A symmetric system without drift is small-time controllable from X iff the rank of the vector space spanned by the family of vector fields f_i together with all their brackets is n at X.*

Checking the Lie algebra rank condition (LARC) on a control system consists in trying to build a basis of the tangent space from a basis (e.g., a P. Hall family) of the free Lie algebra spanned by the control vector fields. An algorithm appears in [46,50].

2.2 Mobile robots: from dynamics to kinematics

Modeling mobile robots with wheels as control systems may be addressed with a differential geometric point of view by considering only the classical hypothesis of "rolling without slipping". Such a modeling provides directly kinematic models of the robots. Nevertheless, the complete chain from motion planning to motion execution requires to consider the ultimate controls that should be applied to the true system. With this point of view, the kinematic model should be derived from the dynamic one. Both view points converge to the same modeling (e.g., [47]) but the later enlightens on practical issues more clearly than the former.

[1] The k-th coordinate of $[f, g]$ is

$$[f, g][k] = \sum_{i=1}^{n} (g[i] \frac{\partial}{\partial x_i} f[k] - f[i] \frac{\partial}{\partial x_i} g[k]).$$

Let us consider two systems: a two-driving wheel mobile robot and a car (in [17] other mechanical structure of mobile robots are considered).

Two-driving wheel mobile robots A classical locomotion system for mobile robot is constituted by two parallel driving wheels, the acceleration of each being controlled by an independent motor. The stability of the platform is insured by castors. The reference point of the robot is the midpoint of the two wheels; its coordinates, with respect to a fixed frame are denoted by (x, y); the main direction of the vehicle is the direction θ of the driving wheels. With ℓ designating the distance between the driving wheels the dynamic model is:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v_1} \\ \dot{v_2} \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(v_1 + v_2)\cos\theta \\ \frac{1}{2}(v_1 + v_2)\sin\theta \\ \frac{1}{\ell}(v_1 - v_2) \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} u_2 \qquad (1)$$

with $|u_1| \leq u_{1,max}$, $|u_2| \leq u_{2,max}$ and v_1 and v_2 as the respective wheel speeds. Of course v_1 and v_2 are also bounded; these bounds appear at this level as "obstacles" to avoid in the 5-dimensional manifold. This 5-dimensional system is not small-time controllable from any point (this is due to the presence of the drift and to the bounds on u_1 and u_2).

By setting $v = \frac{1}{2}(v_1 + v_2)$ and $\omega = \frac{1}{\ell}(v_1 - v_2)$ we get the kinematic model which is expressed as the following 3-dimensional system:

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

The bounds on v_1 and v_2 induce bounds v_{max} and ω_{max} on the new controls v and ω. This system is symmetric without drift; applying the LARC condition shows that it is small-time controllable from everywhere. Notice that v and ω should be C^1.

Car-like robots From the driver's point of view, a car has two controls: the accelerator and the steering wheel. The reference point with coordinates (x, y) is the midpoint of the rear wheels. We assume that the distance between both rear and front axles is 1. We denote w as the speed of the front wheels of the car and ζ as the angle between the front wheels and the main direction θ of the car[2]. Moreover a mechanical constraint imposes $|\zeta| \leq \zeta_{max}$ and consequently a

[2] More precisely, the front wheels are not exactly parallel; we use the average of their angles as the turning angle.

minimum turning radius. Simple computation shows that the dynamic model of the car is:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{w} \\ \dot{\zeta} \end{pmatrix} = \begin{pmatrix} w\cos\zeta\cos\theta \\ w\cos\zeta\sin\theta \\ w\sin\zeta \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} u_2 \tag{3}$$

with $|u_1| \leq u_{1,max}$ and $|u_2| \leq u_{2,max}$. This 5-dimensional system is not small-time controllable from everywhere.

A first simplification consists in considering w as a control; it gives a 4-dimensional system:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\zeta} \end{pmatrix} = \begin{pmatrix} \cos\zeta\cos\theta \\ \cos\zeta\sin\theta \\ \sin\zeta \\ 0 \end{pmatrix} w + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} u_2 \tag{4}$$

This new system is symmetric without drift; applying the LARC condition shows that it is small-time controllable from everywhere. Notice that w should be C^1. Up to some coordinate changes, we may show that this system is equivalent to the kinematic model of a two-driving wheel mobile robot pulling a "trailer" which is the rear axle of the car (see below). The mechanical constraint $|\zeta| \leq \zeta_{max} \leq \frac{\pi}{2}$ appears as an "obstacle" in $\mathbf{R}^2 \times (S^1)^2$.

Let us assume that we do not care about the direction of the front wheels. We may still simplify the model. By setting $v = w\cos\zeta$ and $\omega = w\sin\zeta$ we get a 3-dimensionated control system:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \omega \tag{5}$$

By construction v and ω are C^1 and their values are bounded. This system looks like the kinematic model of the two-driving wheel mobile robot. The main difference lies on the admissible control domains. Here the constraints on v and ω are no longer independent. Indeed, by setting $w_{max} = \sqrt{2}$ and $\zeta_{max} = \frac{\pi}{4}$ we get: $0 \leq |\omega| \leq |v| \leq 1$. This means that the admissible control domain is no longer convex. It remains symmetric; we can still apply the LARC condition to prove that this system is small-time controllable from everywhere. The main difference with the two-driving wheel mobile robot is that the feasible paths of the car should have a curvature lesser than 1.

A last simplification consists in putting $|v| \equiv 1$ and even $v \equiv 1$; by reference to the work in [65] and [22] on the shortest paths in the plane with

bounded curvature such systems will be called Reeds&Shepp's car and Dubins' car respectively (see Souères–Boissonnat's chapter for an overview of recent results on shortest paths for car-like robots). The admissible control domain of Reeds&Shepp's car is symmetric; LARC condition shows that it is small-time controllable from everywhere[3]. Dubins' car is a system with drift; it is locally controllable but not small-time controllable from everywhere; for instance, to go from $(0,0,0)$ to $(1-\cos\epsilon, \sin\epsilon, 0)$ with Dubins car takes at least $2\pi-\epsilon$ unity of time.

The difference between the small-time local controllability of the car of Reeds & Shepp and the local controllability of Dubins' car may be illustrated geometrically. Figure 1 shows the accessibility surfaces in $\mathbf{R}^2 \times S^1$ of both systems for a fixed length of the shortest paths. Such surfaces have been computed from the synthesis of the shortest paths for these systems (see [76,51,15] and Souères–Boissonnat's chapter). In the case of Reeds&Shepp's car, the surface encloses a neighborhood of the origin; in the case of Dubins' car the surface is not connected and it does not enclose any neighborhood of the origin.

2.3 Kinematic model of mobile robots with trailers

Let us now introduce the mobile robot with trailers which has been the canonical example of the work in nonholonomic motion planning; it will be the leading thread of the rest of the presentation.

Figure 2 (left) shows a two-driving wheel mobile robot pulling two trailers; each trailer is hooked up at the middle point of the rear wheels of the previous one. The distance between the reference points of the trailers is assumed to be 1. The kinematic model is defined by the following control system (see [47]) :

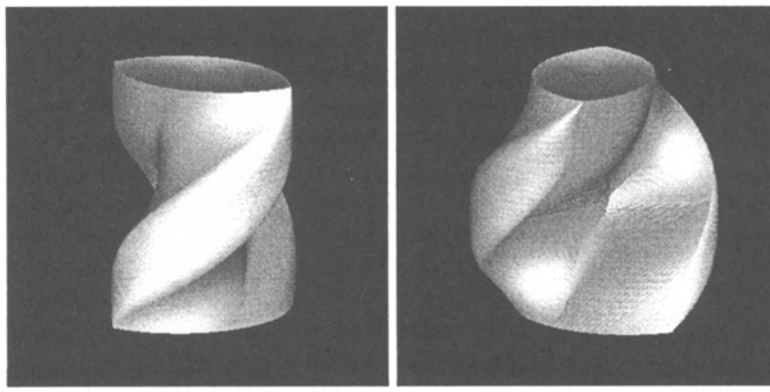
$$\dot{X} = f_1(X)v + f_2(X)\omega \tag{6}$$

with
$X = (x, y, \theta, \varphi_1, \varphi_2)^T$
$f_1(X) = (\cos\theta,\ \sin\theta,\ 0,\ -\sin\varphi_1,\ \sin\varphi_1 - \cos\varphi_1 \sin\varphi_2)^T$ and
$f_2(X) = (0,\ 0,\ 1,\ 1,\ 0)^T$

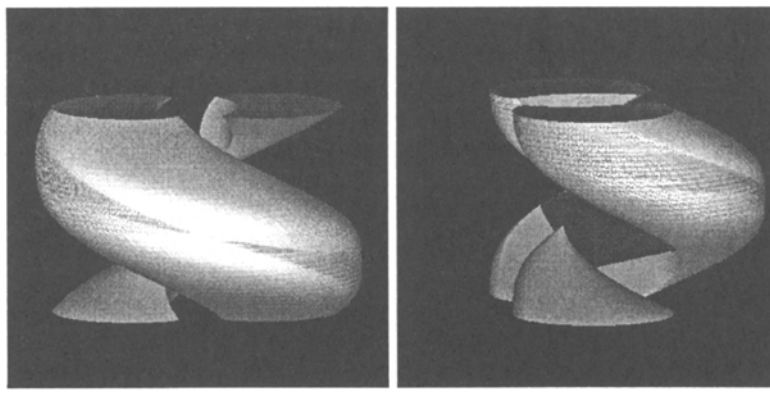
Note that the first body can be viewed as the front wheels of a car; the system then appears as modeling a car-like robot pulling a trailer.

After noticing that $[f_2,[f_1,f_2]] = f_1$, one may check that the family composed of $\{f_1, f_2, [f_1,f_2], [f_1,[f_1,f_2]], [f_1,[f_1,[f_1,f_2]]]\}$ spans the tangent space at every point in $\mathbf{R}^2 \times (S^1)^3$ verifying $\varphi_1 \not\equiv \frac{\pi}{2}$ (regular points). The family $\{f_1, f_2, [f_1,f_2], [f_1,[f_1,f_2]], [f_1,[f_1,[f_1,f_2]]], [f_1,[f_1,[f_1,[f_1,f_2]]]]\}$ spans the tangent space elsewhere (i.e., at singular points). Thanks to the LARC, we conclude that the

[3] A geometric proof appears in [44].



Two points of view of the same Reeds&Shepp's ball



Two points of view of the same Dubins' "ball"

Fig. 1. Accessibility domains by shortest paths of fixed length

system is small-time controllable at any point. Its degree of nonholonomy[4] is 4 at regular points and 5 at singular points. A more general proof of small-time controllability for this system with n trailers appears in [47].

Another hooking system is illustrated in Figure 2 (right). Let us assume that the distance between the middle point of the wheels of a trailer and the hookup of the preceding one is 1. The control system is the same as (6), with

[4] The minimal length of the Lie bracket required to span the tangent space at a point is said to be the degree of nonholonomy of the system at this point. The degree of nonholonomy of the system is the upper bound d of all the degrees of nonholonomy defined locally (see Bellaïche–Jean–Risler's Chapter for details).

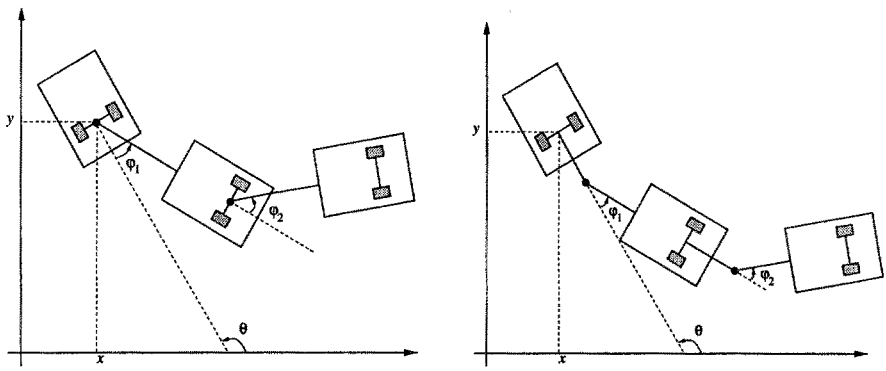


Fig. 2. Two types of mobile robots with trailers.

$f_1(X) = (\cos\theta,\ \sin\theta,\ 0,\ -\sin\varphi_1,\ -\sin\varphi_2\cos\varphi_1+\cos\varphi_2\sin\varphi_1+\sin\varphi_1)^T$ and $f_2(X) = (0,\ 0,\ 1,\ -1-\cos\varphi_1,\ \sin\varphi_1\sin\varphi_2+\cos\varphi_1\cos\varphi_2+\cos\varphi_1)^T$

The family $\{f_1, f_2, [f_1, f_2], [f_1, [f_1, f_2]], [f_2, [f_1, f_2]]\}$ spans the tangent space at every point in $\mathbf{R}^2 \times (S^1)^3$ verifying $\varphi_1 \not\equiv \pi$, $\varphi_2 \not\equiv \pi$ and $\varphi_1 \not\equiv \varphi_2$ (regular points). The degree of nonholonomy is then 3 at regular points. The family $\{f_1, f_2, [f_1, f_2], [f_1, [f_1, f_2]], [f_1, [f_1, [f_1, f_2]]]\}$ spans the tangent space at points verifying $\varphi_1 \equiv \varphi_2$. The degree of nonholonomy at these points is then 4. When $\varphi_1 \equiv \pi$ or $\varphi_2 \equiv \pi$ the system is no more controllable; this is a special case of mechanical singularities.

2.4 Admissible paths and trajectories

Constrained paths and trajectories Let $\mathcal{CS}$ be the configuration space of some mobile robot (i.e., the minimal number of parameters locating the whole system in its environment). In the sequel a *trajectory* is a continuous function from some real interval $[0, T]$ in $\mathcal{CS}$. An *admissible* trajectory is a solution of the differential system corresponding to the kinematic model of the mobile robot (including the control constraints), with some initial and final given conditions. A *path* is the image of a trajectory in $\mathcal{CS}$. An *admissible* path is the image of an admissible trajectory.

The difference between the various kinematic models of the mobile robots considered in this presentation only concerns their control domains (Figure 3). It clearly appears that admissible paths for Dubins' car are admissible for Reeds&Shepp's car (the converse is false); admissible paths for Reeds&Shepp's car are admissible for the car-like robot (the converse is true); admissible paths for the car-like robot are admissible for the two-driving wheel mobile robot (the converse is false).

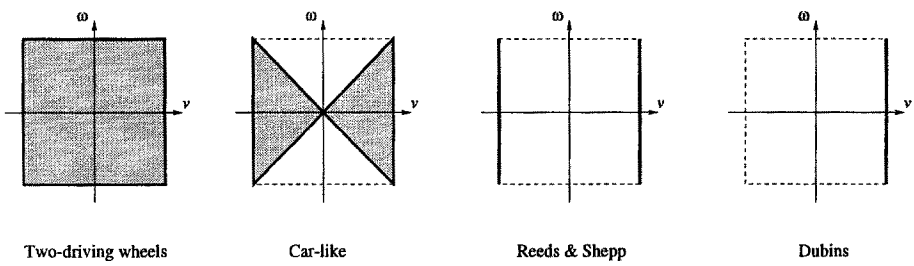


Fig. 3. Kinematic mobile robot models: four types of control domains.

Remark 1: Due to the constraint $|\omega| < |v|$, the admissible paths for the car-like, Reeds&Shepp's and Dubins' robots have their curvature upper bounded by 1 everywhere. As a converse any curve with curvature upper bounded by 1 is an admissible path (i.e., it is possible to compute an admissible trajectory from it).

Remark 2: This geometric constraint can be taken into account by considering the four-dimensionated control system (4) with $|\zeta| \leq \frac{\pi}{4}$; the inequality constraint on the controls for the 3-dimensionated system is then transformed into a geometric constraint on the state variable ζ. Therefore the original control constraint $|\omega| < |v|$ arising in system (5) can be addressed by applying "obstacle" avoidance techniques to the system (4).

From paths to trajectories The goal of nonholonomic motion planning is to provide *collision-free admissible paths* in the configuration space of the mobile robot system. Obstacle avoidance imposes a geometric point of view that dominates the various approaches addressing the problem. The motion planners compute paths which have to be transformed into trajectories.

In almost all applications, a black-box module allows to control directly the linear and angular velocities of the mobile robot. Velocities and accelerations are of course submitted bounds.

The more the kinematic model of the robot is simplified, the more the transformation of the path into a trajectory should be elaborated. Let us consider for instance an elementary path consisting of an arc of a circle followed by a tangent straight line segment. Due to the discontinuity of the curvature of the path at the tangent point, a two driving-wheel mobile robot should stop at this point; the resulting motion is clearly not satisfactory. This critical point may be overcome by "smoothing" the path before computing the trajectory. For instance clothoids and involutes of a circle are curves that account for the dynamic model of a two driving-wheel mobile robot: they correspond to bang-

bang controls for the system (1) [35]; they may be used to smooth elementary paths [25].

Transforming an admissible path into an admissible trajectory is a classical problem which has been investigated in robotics community mainly through the study of manipulators (e.g., [67] for a survey of various approaches). Formal solutions exist (e.g., [75] for an approach using optimal control); they apply to our problem. Nevertheless, their practical programming tread on delicate numerical computations [40].

On the other hand, some approaches address simultaneously the geometric constraints of obstacle avoidance, the kinematic and the dynamic ones; this is the so-called "kinodynamic planning problem" (e.g., [20,21,66]). These methods consist in exploring the phase space (i.e., the tangent bundle associated to the configuration space of the system) by means of graph search and discretization techniques. In general, such algorithms provide approximated solutions (with the exception of one and two dimensional cases [62,19]) and are time-consuming. Only few of them report results dealing with obstacle avoidance for nonholonomic mobile robots (e.g., [28]).

The following developments deal with nonholonomic *path* planning.

3 Path planning and small-time controllability

Path planning raises two problems: the first one addresses the *existence* of a collision-free admissible path (this is the decision problem) while the second one addresses the *computation* of such a path (this is the complete problem).

The results overviewed in this section show that the decision problem is solved for any small-time controllable system; even if approximated algorithms exist to solve the complete problem, the exact solutions deal only with some special classes of small-time controllable systems.

We may illustrate these statements with the mobile robot examples introduced in the previous section:

- Dubins' robot: this is the simplest example of a system which is locally controllable and not small-time controllable. For this system, the decision problem is solved when the robot is reduced to a point [27]. An approximated solution of the complete problem exists [34]; exact solutions exist for a special class of environments consisting of *moderated* obstacles (moderated obstacles are generalized polygons whose boundaries are admissible paths for Dubins' robot) [2,13]. Notice that the decision problem is still open when the robot is a polygon.

- Reeds&Shepp's, car-like and two-driving wheel robots: these systems are small-time controllable. We will see below that exact solutions exist for both problems.
- Mobile robots with trailers: the two systems considered in the previous section are generic of the class of small-time controllable systems. For both of them the decision problem is solved. For the system appearing in Figure 2 (left) we will see that the complete problem is solved; it remains open for the system in Figure 2 (right).

Small-time controllability (Definition 1) has been introduced with a control theory perspective. To make this definition operational for path planning, we should translate it in purely geometric terms.

Let us consider a small-time controllable system, with $\mathcal{U}$ a class of control functions taking their values in some compact domain $\mathcal{K}$ of $\mathbf{R}^m$. We assume that the system is symmetric[5]. As a consequence, for any admissible path between two configurations X_1 and X_2, there are two types of admissible trajectories: the first ones go from X_1 to X_2, the second ones go from X_2 to X_1.

Let X be some given configuration. For a fixed time T, let $\mathcal{R}each_X(T)$ be the set of configurations reachable from X by an admissible trajectory before the time T. $\mathcal{K}$ being compact, $\mathcal{R}each_X(T)$ tends to $\{X\}$ when T tends to 0.

Because the system is small-time controllable, $\mathcal{R}each_X(T)$ contains a neighborhood of X. We assume that the configuration space is equipped with a (Riemannian) metric: any neighborhood of a point contains a ball centered at this point with a strictly positive radius. Then there exists a positive real number η such that the ball $B(X,\eta)$ centered at X with radius η is included in $\mathcal{R}each_X(T)$.

Now, let us consider a (not necessarily admissible) collision-free path γ with finite length linking two configurations X_{start} and X_{goal}. γ being compact, it is possible to define the clearance ϵ of the path as the minimum distance of γ to the obstacles[6]. ϵ is strictly positive. Then for any X on γ, there exists $T_X > 0$ such that $\mathcal{R}each_X(T_X)$ does not intersect any obstacle. Let η_{T_X} be the radius of the ball centered at X whose points are all reachable from X by admissible trajectories that do not escape $\mathcal{R}each_X(T_X)$. The set of all the balls $B(X,\eta_{T_X})$, $X \in \gamma$, constitutes a covering of γ. γ being compact, it is possible to get a finite sequence of configurations $(X_i)_{1\leq i\leq k}$ (with $X_1 = X_{start}$, $X_k = X_{goal}$), such that the balls $B(X_i,\eta_{T_{X_i}})$ cover γ.

Consider a point $Y_{i,i+1}$ lying on γ and in $B(X_i,\eta_{T_{X_i}}) \cap B(X_{i+1},\eta_{T_{X_{i+1}}})$. Between X_i and $Y_{i,i+1}$ (respectively X_{i+1} and $Y_{i,i+1}$) there is an admissible

[5] Notice that, with the exception of Dubins' robot, all the mobile robots introduced in the previous section are symmetric.

[6] We consider that a configuration where the robot touches an obstacle is not collision-free.

trajectory (and then an admissible path) that does not escape $\mathcal{R}each_{X_i}(T_{X_i})$ (respectively $\mathcal{R}each_{X_{i+1}}(T_{X_{i+1}})$). Then there is an admissible path between X_i and X_{i+1} that does not escape $\mathcal{R}each_{X_i}(T_{X_i}) \cup \mathcal{R}each_{X_{i+1}}(T_{X_{i+1}})$; this path is then collision-free. The sequence $(X_i)_{1 \leq i \leq k}$ is finite and we can conclude that there exists a collision-free admissible path between X_{start} and X_{goal}.

Theorem 3.1. *For symmetric small-time controllable systems the existence of an admissible collision-free path between two given configurations is equivalent to the existence of any collision-free path between these configurations.*

Remark 3: We have tried to reduce the hypothesis required by the proof to a minimum. They are realistic for practical applications. For instance the compactness of $\mathcal{K}$ holds for all the mobile robots considered in this presentation. Moreover we assume that we are looking for admissible paths without contact with the obstacles: this hypothesis is realistic in mobile robotics (it does not hold any more for manipulation problems). On the other hand we suggest that two configurations belonging to the same connected component of the collision-free path can be linked by a finite length path; this hypothesis does not hold for any space (e.g., think to space with a fractal structure); nevertheless it holds for realistic workspaces where the obstacles are compact, where their shape is simple (e.g., semi-algebraic) and where their number is finite.

Consequence 1: Theorem 3.1 shows that the decision problem of motion planning for a symmetric small-time controllable nonholonomic system is the same as the decision problem for the holonomic associated one (i.e., when the kinematics constraints are ignored): it is decidable. Notice that deciding whether some general symmetric system is small-time controllable (from everywhere) can be done by a only semi-decidable procedure [50]. The combinatorial complexity of the problem is addressed in [77]. Explicit bounds of complexity have been recently provided for polynomial systems in the plane (see [68] and references therein).

Consequence 2: Theorem 3.1 suggests an approach to solve the complete problem. First, one may plan a collision-free path (by means of any standard methods applying to the classical piano mover problem); then, one approximates this first path by a finite sequence of admissible and collision-free ones. This idea is at the origin of a nonholonomic path planner which is presented below (Section 5.3). It requires effective procedures to steer a nonholonomic system from a configuration to another. The problem has been first attacked by ignoring the presence of obstacles (Section 4); numerous methods have been mainly developed within the control theory community; most of them account only for local controllability. Nevertheless, the planning scheme suggested by Theorem 3.1 requires steering methods that accounts for small-time controllability

(i.e., not only for local controllability). In Section 5.1 we introduce a topological property which is required by steering methods in order to apply the planning scheme. We show that some among those presented in Section 4 verify this property, another one does not, and finally a third one may be extended to guaranty the property.

4 Steering methods

What we call a steering method is an algorithm that solves the path planning problem without taking into account the geometric constraints on the state. Even in the absence of obstacles, computing an admissible path between two configurations of a nonholonomic system is not an easy task. Today there is no algorithm that guarantees any nonholonomic system to reach an accessible goal exactly. In this section we present the main approaches which have been applied to mobile robotics.

4.1 From vector fields to effective paths

The concepts from differential geometry that we want to introduce here are thoroughly studied in [79,90,80,81] and in Bellaïche–Jean–Risler's Chapter. They give a combinatorial and geometric point of view of the path planning problem.

Choose a point X on a manifold and a vector field f defined around this point. There is exactly one path $\gamma(\tau)$ starting at this point and following f. That is, it satisfies $\gamma(0) = X$ and $\dot{\gamma}(\tau) = f(\gamma(\tau))$. One defines the exponential of f at point X to be the point $\gamma(1)$ denoted by $e^f.X$. Therefore e^f appears as an operation on the manifold, meaning "slide from the given point along the vector field f for unit time". This is a diffeomorphism. With α being a real number, applying $e^{\alpha f}$ amounts to follow f for a time α. In the same way, applying e^{f+g} is equivalent to follow $f + g$ for unit time.

It remains that, whenever $[f, g] \neq 0$, following directly $\alpha f + \beta g$ or following first αf, then βg, are no longer equivalent. Intuitively, the bracket $[f, g]$ measures the variation of g along the paths of f; in some sense, the vector field g we follow in $\alpha f + \beta g$ is not the same as the vector field g we follow after having followed αf first (indeed g is not evaluated at the same points in both cases).

Assume that $f_1, \ldots, f_n$ are vector fields defined in a neighborhood $\mathcal{N}$ of a point X such that at each point of $\mathcal{N}$, $\{f_1, \ldots, f_n\}$ constitutes a basis of the tangent space. Then there is a smaller neighborhood of X on which the maps $(\alpha_1, \ldots, \alpha_n) \mapsto e^{\alpha_1 f_1 + \cdots + \alpha_n f_n} \cdot X$ and $(\alpha_1, \ldots, \alpha_n) \mapsto e^{\alpha_n f_n} \ldots e^{\alpha_1 f_1} \cdot X$ are two coordinate systems, called the first and the second normal coordinate system associated to $\{f_1, \ldots, f_n\}$.

The Campbell-Baker-Hausdorff-Dynkin formula states precisely the difference between the two systems: for a sufficiently small τ, one has:

$$e^{\tau f} \cdot e^{\tau g} = e^{\tau f + \tau g - \frac{1}{2}\tau^2[f,g] + \tau^2 \epsilon(\tau)}$$

where $\epsilon(\tau) \to 0$ when $\tau \to 0$.

Actually, the whole formula as proved in [90] gives an explicit form for the ϵ function. More precisely, ϵ yields a formal series whose coefficients c_k of τ^k are combinations of brackets of degree k,[7] i.e.

$$\tau^2 \epsilon(\tau) = \sum_{k=3}^{\infty} \tau^k c_k$$

Roughly speaking, the Campbell-Baker-Hausdorff-Dynkin formula tells us how a small-time nonholonomic system can reach *any* point in a neighborhood of a starting point. This formula is the hard core of the local controllability concept. It yields methods for *explicitly computing admissible paths* in a neighborhood of a point.

4.2 Nilpotent systems and nilpotentization

One method among the very first ones has been defined by Lafferiere and Sussmann [39] in the context of nilpotent system. A control system is nilpotent as soon as the Lie brackets of the control vector fields vanish from some given length.

For small-time controllable nilpotent systems it is possible to compute a basis $\mathcal{B}$ of the Control Lie Algebra $LA(\Delta)$ from a Philipp Hall family (see for instance [46]). The method assumes that a holonomic path γ is given. If we express locally this path on $\mathcal{B}$, i.e., if we write the tangent vector $\dot{\gamma}(t)$ as a linear combination of vectors in $\mathcal{B}(\gamma(t))$, the resulting coefficients define a control that steers the holonomic system along γ. Because the system is nilpotent, each exponential of Lie bracket can be developed *exactly* as a finite combination of the control vector fields: such an operation can be done by using the Campbell-Baker-Hausdorff-Dynkin formula above. An introduction to this machinery through the example of a car-like robot appears in [48]. It is then possible to compute an admissible and *piecewise constant* control u for the nonholonomic system that steers the system *exactly* to the goal.

For a general system, Lafferiere and Sussmann reason as if the system were nilpotent of order k. In this case, the synthesized path deviates from the goal. Nevertheless, thanks to a topological property, the basic method may be used

[7] As an example the degree of $[[f,g],[f,[g,[f,g]]]]$ is 6.

in an iterated algorithm that produces a path ending as close to the goal as wanted.

In [33], Jacob gives an account of Lafferiere and Sussmann's strategy by using another coordinate system. This system is built from a Lyndon basis of the free Lie algebra [93] instead of a P. Hall basis. This choice reduces the number of pieces of the solution.

In [11], Bellaïche *et al* apply the nilpotentization techniques developed in [10] (see also [31]). They show how to transform any controllable system into a canonical form corresponding to a nilpotent system approximating the original one. Its special triangular form allows to apply sinusoidal inputs (see below) to steer the system locally. Moreover, it is possible to derive from the proposed canonical form an estimation of the metrics induced by the shortest feasible paths. This estimation holds at regular points (as in [92]) as well as at singular points. These results are critical to evaluate the combinatorial complexity of the approximation of holonomic paths by a sequence of admissible ones (see Section 5.7).

The mobile robots considered in this presentation are not nilpotent[8]. A nilpotentization of this system appear in [39]. We conclude this section by the nilpotentization of a mobile robot pulling a trailer [11].

Example: Let us consider the control system 6:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \\ -\sin\varphi \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} u_2$$
$$= \quad f_1 u_1 \quad + \quad f_2 u_2$$

where (x, y) defines the position of the mobile robot, θ its direction and φ the angle of the trailer with respect to the mobile robot.

The coordinates of vector fields $f_3 = [f_1, f_2]$ and $f_4 = [f_1, [f_1, f_2]]$ are respectively:

$$f_3 = \begin{pmatrix} \sin\theta \\ -\cos\theta \\ 0 \\ \cos\varphi \end{pmatrix} \qquad f_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

We check easily that $\{f_1, f_2, f_3, f_4\}$ is a basis of the tangent space at every point of the manifold $\mathbf{R}^2 \times (S^1)^2$. Let $X_0 = (x_0, y_0, \theta_0, \varphi_0)$ and $X = (x, y, \theta, \varphi)$ be

[8] Consider the system (2); let us denote f_1 and f_2 the two vector fields corresponding to a straight line motion and a rotation respectively. By setting $ad_f(g) = [f, g]$, we check that $ad_{f_2}^{2m}(f_1) = (-1)^m f_1 \neq 0$.

two points of the manifold. By writing $\Delta x = x - x_0$, $\Delta y = y - y_0$, $\Delta\theta = \theta - \theta_0$ and $\Delta\varphi = \varphi - \varphi_0$, the coordinates (y_1, y_2, y_3, y_4) of X in the chart attached to X_0 with the basis $\{f_1, f_2, f_3, f_4\}(X_0)$ are:

$$\begin{aligned} y_1 &= \cos\theta_0 \Delta x + \sin\theta_0 \Delta y \\ y_2 &= \Delta\theta \\ y_3 &= \sin\theta_0 \Delta x - \cos\theta_0 \Delta y \\ y_4 &= \sin(\varphi_0 - \theta_0)\Delta x + \cos(\varphi_0 - \theta_0)\Delta y - \Delta\theta + \Delta\varphi \end{aligned}$$

The goal of the following computations is to provide a new coordinate system (z_1, z_2, z_3, z_4) at X_0 such that:

- $((f_i)z_k)(X_0) = \delta^i_k$,
- there exists i and j such that $((f_i.f_j)z_3)(X_0) \neq 0$,
- for any i and j, $((f_i.f_j)z_4)(X_0) = 0$, and
- there exists i, j and k such that $((f_h.f_i.f_j)z_4)(X_0) \neq 0$

with $h, i, j \in \{1, 2\}$ and $k \in \{1, 2, 3, 4\}$; $\delta^i_k = 1$ iff $i = k$; (f) designates the differential operator associated to the vector field f; $(f.g)$ is the product of the corresponding differential operators. Such coordinates are called *privileged coordinates.*

One may check that $((f_i)y_k)(X_0) = \delta^i_k$ for $i \in \{1, 2\}$ and $k \in \{1, 2, 3, 4\}$. Moreover $((f_1)^2 y_3)(X_0) = ((f_2)^2 y_3)(X_0) = 0$ and $((f_2.f_1)y_3)(X_0) = 1$. Now, it appears that $((f_1)^2 y_4)(X_0) = \sin\varphi_0 \cos\varphi_0$; then (y_1, y_2, y_3, y_4) is not a privileged coordinate system if $\sin\varphi_0 \cos\varphi_0 \neq 0$.

One gets privileged coordinates by keeping

$$z_1 = y_1, \quad z_2 = y_2, \quad z_3 = y_3$$

and taking

$$z_4 = y_4 - \frac{1}{2}\sin\varphi_0 \cos\varphi_0 y_1^2.$$

In such coordinates, we have

$$f_1 = \begin{pmatrix} \cos z_2 \\ 0 \\ -\sin z_2 \\ F(z_1, z_2, z_3, z_4) \end{pmatrix} \qquad f_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{7}$$

where

$$\begin{aligned} F(z_1, z_2, z_3, z_4) &= -z_1(\cos z_2 \sin 2\varphi_0)/2 + \sin(\varphi_0 + z_2) \\ &- \sin\big(\varphi_0 - z_1 \sin\varphi_0 + z_1^2(\sin 2\varphi_0)/4 + z_2 + z_3 \cos\varphi_0 + z_4\big). \end{aligned}$$

The nilpotent approximation is obtained by taking in the Taylor expansions of (7) the terms of homogeneous degree $w_i - 1$ for the i-th coordinate where w_i is the degree of the vector field f_i (i.e., $w_1 = w_2 = 1, w_3 = 2, w_4 = 3$). We get

$$\widehat{f}_1 = \begin{pmatrix} 1 \\ 0 \\ -z_2 \\ \widehat{F}(z_1, z_2, z_3) \end{pmatrix} \quad \widehat{f}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

where

$$\widehat{F}(z_1, z_2, z_3) = -z_1^2(\sin\varphi_0 \cos 2\varphi_0)/2 - z_1 z_2 \sin^2\varphi_0 - z_3 \cos^2\varphi_0.$$

It is easy to check that this new system is nilpotent of order 3.

4.3 Steering chained form systems

At the same time as Lafferiere and Sussmann work, Murray and Sastry explored in [58,59] the use of sinusoidal inputs to steer certain nonholonomic systems: the class of systems which can be converted into a chained form. A chained system has the following form:

$$\begin{aligned} \dot{x}_1 &= v \\ \dot{x}_2 &= f_2(x_1)v \\ \dot{x}_3 &= f_3(x_1, x_2)v \\ &\vdots \\ \dot{x}_p &= f_p(x_1, \ldots, x_p)v \end{aligned}$$

with $x_i \in \mathbf{R}^{m_i}$ and $\sum_i m_i = n$.

Because of this special form, there exists simple sinusoidal control that may be used for generating motions affecting the i^{th} set of coordinates while leaving the previous sets of coordinates unchanged. The algorithm then is:

1. Steer x_1 to the desired value using any input and ignoring the evolutions of the x_i's $(1 < i)$,
2. Using sinusoids at integrally related frequencies, iteratively find the inputs steering the x_i's without changing the x_j's, $j < i$.

Extensions [86] by Tilbury and Sastry allow to use sinusoidal control to steer all the coordinates at once for systems with two inputs. They show also how polynomial controls may be used to this end. Moreover Monaco and Normand-Cyrot show that multirate controls (i.e., piece-wise constant controls) provide an exact steering method for chained form systems [57].

Even if a system is not triangular, it may be possible to transform it into a triangular one by feedback transformations (see [59,60]). Moreover, notice that the nilpotentization techniques introduced in the previous section leads to approximated systems which are in chained form.

Example: Let us consider our canonical example of a mobile robot with two trailers (Figure 2, left). The clever idea which enables the transformation of the system into a chained form was to consider a frame attached to the last trailer rather than attached to the robot [86]. Denoting by θ_1 and θ_2 the angle of the trailers, and by x_2 and y_2 the coordinates of the middle point of the last trailer, the system (6) may be re-written as:

$$\begin{cases} \dot{x} = cos\theta_2 cos(\theta_1 - \theta_2) cos(\theta_0 - \theta_1) u_1 \\ \dot{y} = sin\theta_2 cos(\theta_1 - \theta_2) cos(\theta_0 - \theta_1) u_1 \\ \dot{\theta}_0 = u_2 \\ \dot{\theta}_1 = \sin(\theta_0 - \theta_1) u_1 \\ \dot{\theta}_2 = \sin(\theta_1 - \theta_2) cos(\theta_0 - \theta_1) u_1 \end{cases}$$

Let us consider the following change of coordinates:

$$\begin{cases} z_1 = x \\ z_2 = \frac{1}{cos^4\theta_2} \cdot \frac{tan(\theta_0 - \theta_1)}{cos(\theta_1 - \theta_2)} \times (1 + tan^2(\theta_1 - \theta_2)) \\ \quad + \frac{1}{cos^4\theta_2} \times tan(\theta_1 - \theta_2)(3tan(\theta_1 - \theta_2) tan\theta_2 \; - \; (1 - tan^2(\theta_1 - \theta_2))) \\ z_3 = \frac{tan(\theta_1 - \theta_2)}{cos^3\theta_2} \\ z_4 = tan\theta_2 \\ z_5 = y \end{cases}$$

This transformation is a local diffeomorphism around configurations for which the angle between bodies are not equal to $\frac{\pi}{2}$. In this new coordinates, the kinematic model of the system has the following chained form:

$$\begin{cases} \dot{z}_1 = v_1 \\ \dot{z}_2 = v_2 \\ \dot{z}_3 = z_2 . v_1 \\ \dot{z}_4 = z_3 . v_1 \\ \dot{z}_5 = z_4 . v_1 \end{cases} \tag{8}$$

Notice that Sordalen generalizes this result by providing a conversion of the car with an arbitrary number of trailers into a chained form [78].

Sinusoidal inputs: Let us consider the following inputs [86]:

$$\begin{cases} v_1(t) = a_0 + a_1 \sin \omega t \\ v_2(t) = b_0 + b_1 \cos \omega t + b_2 \cos 2\omega t + b_3 \cos 3\omega t \end{cases} \tag{9}$$

Let Z^{start} be a starting configuration. Equations (8) are integrable. Then each $z_i(T)$ can be computed from the five coordinates of Z^{start} and the six parameters $(a_0, a_1, b_0, b_1, b_2, b_3)$. For a given $a_1 \neq 0$ and a given configuration Z^{start}, Tilbury *et al* show that the function computing $Z(T)$ from $(a_0, b_0, b_1, b_2, b_3)$ is a C^1 diffeomorphism at the origin; then the system is invertible and the parameters $(a_0, b_0, b_1, b_2, b_3)$ can be computed from the coordinates of two configurations Z^{start} and Z^{goal}. The system inversion can be done with the help of any symbolic computation software. The corresponding sinusoidal inputs steer the system from Z^{start} to Z^{goal}.

The shape of the path only depends on the parameter a_1. Figure 4 from [71] illustrates this dependence. Moreover the shape of the paths is not invariant by rotation (i.e., it depends on the variables θ^{start} and θ^{goal} and not only on the difference $(\theta^{start} - \theta^{goal})$).

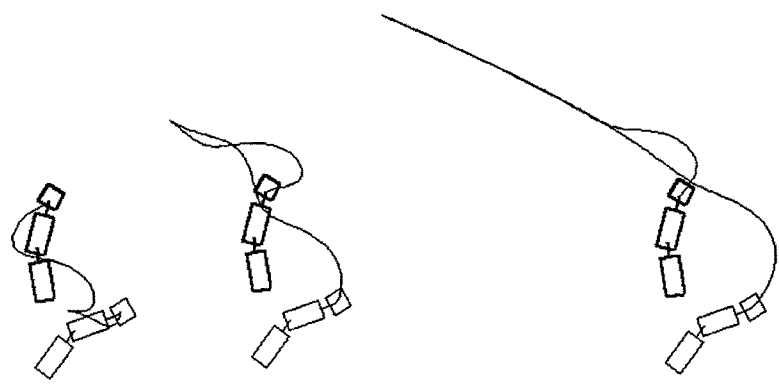


Fig. 4. Three paths solving the same problem with three values of a_1: -30, 70, 110

Polynomial inputs: Another steering method is also proposed in [86]. The polynomial inputs:

$$\begin{cases} v_1(t) = 1 \\ v_2(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 \end{cases}$$

steer the system from any configuration Z^{start} to any Z^{goal} verifying $z_1^{goal} \neq z_1^{start}$. In this case T should be equal to $|z_1^{goal} - z_1^{start}|$. As for the case of the sinusoidal inputs, the system can be inverted by symbolic computation. To reach configurations such that $z_1^{goal} = z_1^{start}$ it is sufficient to choose an intermediate configuration respecting the inequality and to apply the steering method twice.

Extensions: The previous steering methods deal with two-input chained form systems. In [16] Bushnell, Tilbury and Sastry extend these results to three-input nonholonomic systems with the fire-truck system as a canonical example[9]. They give sufficient conditions to convert such a system to two-chain, single generator chained forms. Then they show that multirate digital controls, sinusoidal inputs and polynomial inputs may be used as steering methods.

4.4 Steering flat systems

The concept of flatness has been introduced by Fliess, Lévine, Martin and Rouchon [26,63].

A flat system is a system such that there exists a finite set of variables y_i differentially independent which appear as differential functions of the system variables (state variables and inputs) and of a finite number of their derivatives, each system variable being itself a function of the y_i's and of a finite number of their derivatives. The variables y_i's are called the linearizing outputs of the system.

Example: In [63] Rouchon *et al* show that mobile robots with trailers are flat as soon as the trailers are hitched to the middle point of the wheels of the previous ones. The proof is based on the same idea allowing to transform the system into a chained form: it consists in modeling the system by starting from the last trailer.

Let us consider the system (6) (Figure 2, left). Let us denote the coordinates of the robot and the two trailers by (x, y, θ), (x_1, y_1, θ_1) and (x_2, y_2, θ_2) respectively. Remind that the distance between the reference points of the bodies is 1. The holonomic equations allow to compute x, y, x_1 and y_1 from x_2, y_2, θ_1 and θ_2:

$$x_1 = x_2 + \cos\theta_2 \qquad x = x_2 + \cos\theta_2 + \cos\theta_1$$
$$y_1 = y_2 + \sin\theta_2 \qquad y = y_2 + \sin\theta_2 + \sin\theta_1$$

The rolling without slipping conditions lead to three nonholonomic equations $\dot{x}_i \sin\theta_i - \dot{y}_i \cos\theta_i = 0$ allowing to compute θ_2 (resp. θ_1 and θ) from $(\dot{x_2}, \dot{y_2})$ (resp. $(\ddot{x_2}, \ddot{y_2})$ and $(\dddot{x_2}, \dddot{y_2})$). Finally the controls v and ω are given by $v = \frac{\dot{x}}{\cos\theta}$ (or $v = \frac{\dot{y}}{\sin\theta}$) and $\omega = \dot{\theta}$.

Therefore any variable of the system can be computed from x_2 and y_2 and their derivatives. The system is flat with x_2 and y_2 as linearizing outputs.

[9] The fire-truck system is a car-like robot (two inputs) with one trailer whose direction of the wheels is controllable (third input).

A steering method: Let us consider a path γ_2 followed by the reference point P_2 of the second trailer (Figure 5). γ_2 is parametrized by arc length s_2. Let us assume that γ_2 is sufficiently smooth, i.e., $\frac{d}{ds_2}P_2$ is defined everywhere and the curvature κ_2 can be differentiated at least once. The point P_1 belongs to the tangent to γ_2 at P_2 and $P_1 = P_2 + \tau_2$, with τ_2 the unitary tangent vector to γ_2. Differentiating this relation w.r.t. s_2 leads to $\frac{d}{ds_2}P_1 = \tau_2 + \kappa_2\nu_2$ with ν_2 the unitary vector orthogonal to τ_2. The angle of the first trailer is then $\theta_1 = \theta_2 + atan(\kappa_2)$. We then deduce the path γ_1 followed by the first trailer. Parametrizing γ_1 with s_1 defined by $ds_1 = (1+\kappa_2)^{\frac{1}{2}}ds_2$ leads to

$$\kappa_1 = (1+\kappa_2)^{-\frac{1}{2}}(\kappa_2 + (1+\kappa_2)^{-\frac{1}{2}})\frac{d}{ds_2}\kappa_2$$

Applying the same geometric construction from P_1 we can compute the path γ followed by the robot when the second trailer follows γ_2. The only required condition is the existence of $\frac{d^2}{ds_2^2}\kappa_2$; moreover the relative angles φ_1 and φ_2 should belong to $]-\frac{\pi}{2}, \frac{\pi}{2}[$ (see [26] for details).

Two configurations X^{start} and X^{goal} being given, one computes geometrically the values of κ^{start}, κ_1^{start}, κ_2^{start}, κ^{goal}, κ_1^{goal} and κ_2^{goal}; each of them being a function of κ_2 and its derivative, it is straightforward to compute γ_2 satisfying such initial and final conditions (e.g., by using polynomial curves).

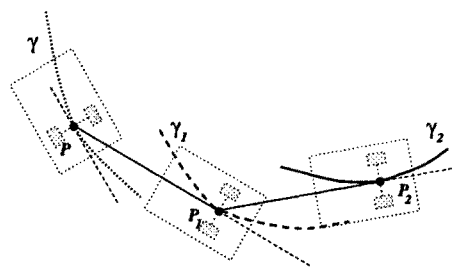


Fig. 5. Geometric construction of P_1 (resp. P) path from P_2 (resp. P_1) path.

Remark: Because the curvature of γ_2 should be defined everywhere, the method can not provide any cusp point; nevertheless such points are required in some situations like the parking task; in that case, Rouchon *et al* enter the cusp point by hand [63]. We will see below how to overcome this difficulty.

Flatness conditions: In cite [64], Rouchon gives conditions to check whether a system is flat. Among them there is a necessary and sufficient condition for

two-input driftless systems: it regards the rank of the various vector space Δ^k iteratively defined by $\Delta_0 = span\{f_1, f_2\}$, $\Delta_1 = span\{f_1, f_2, [f_1, f_2]\}$ and $\Delta_{i+1} = \Delta_0 + [\Delta_i, \Delta_i]$ with $[\Delta_i, \Delta_i] = span\{[f, g]\ ,\ f \in \Delta_i,\ g \in \Delta_i\}$. A system with two inputs is flat iff $rank(\Delta_i) = 2 + i$.

Let us apply this condition to the mobile robot system with two trailers. According to the computations presented in Section 2.3:

- for the case shown in Figure 2 (left), we get:

$$rank(\Delta_0) = 2,\ \ rank(\Delta_1) = 3,\ \ rank(\Delta_2) = 4 \text{ and } rank(\Delta_3) = 5$$

  the system is flat.
- for the case shown in Figure 2 (right), we get:

$$rank(\Delta_0) = 2,\ \ rank(\Delta_1) = 3 \text{ and } rank(\Delta_2) = 5$$

  the system is not flat.
- for the same case shown in Figure 2 (right) but with only one trailer, one can check that:

$$rank(\Delta_0) = 2,\ \ rank(\Delta_1) = 3 \text{ and } rank(\Delta_2) = 4$$

  the system is flat.

We have seen that the linearizing outputs in the first case are the coordinates of the reference point of the second trailer. In the last case, the linearizing outputs are more difficult to translate into geometric terms (see [63]). Notice that there is no general method to compute the linearizing outputs when the system is flat.

4.5 Steering with optimal control

Optimal length paths have been at the origin of the very first nonholonomic motion planners for car-like mobile robots (see for instance [48,43] and below). Nevertheless, today the only existing results allowing to compute optimal paths for nonholonomic systems have been obtained for the car-like systems (see Souères–Boissonnat's Chapter). For general systems, the only possibility is to call on numerical methods.

We sketch here the method developed by Fernandez, Li and Gurvitz in [24].

Let us consider a dynamical system: $\dot{X} = B(X)u$ together with a cost function $J = \int_0^T < u(\tau), u(\tau) > d\tau$. Both starting and goal configurations being given, the optimization problem is to find the control law (if any) that steers the system from the starting configuration to the goal in time T by

minimizing the cost function J. The path corresponding to an optimal control law is said to be an optimal path.

Let us consider a continuous and piecewise C^1 control law u defined over $[0,T]$. We denote by $\tilde{u}$ the periodic extension of u over $\mathbf{R}$. We may write $\tilde{u}$ in terms of a Fourier basis:

$$\tilde{u} = \sum_{k=0}^{\infty} (\alpha_k e^{i\frac{2k\pi t}{T}} + \beta_k e^{-i\frac{2k\pi t}{T}})$$

We then approximate $\tilde{u}$ by truncating its expansion up to some rank N. The new control law $\hat{u}$ is then defined by N real numbers[10] $\hat{u} = \sum_{k=1}^{N} \alpha_k e_k$, $e_k \in \{e^{i\frac{2p\pi}{T}},\ p \in \mathbf{Z}\}$. The choice of the reals α_k being given, the point $X(T)$ reached after a time T with the control law $\hat{u}$ appears as a function $f(\alpha)$ from $\mathbf{R}^N$ to $\mathbf{R}^n$.

Now, we get a new cost function

$$\hat{J}(\alpha) = \sum_{k=1}^{N} |\alpha_k|^2 + \gamma \|X(T) - X_{goal}\|^2.$$

The new optimization problem becomes: for a fixed time T, an initial point X_{init} and a final point X_{goal} , find $\alpha \in \mathbf{R}^N$ such as

$$\hat{J}(\alpha) = \sum_{k=1}^{N} |\alpha_k|^2 + \gamma \|f(\alpha) - X_{goal}\|^2$$

is minimum.

One proves (*e.g.*, see [24]) that the solutions of the new finite-dimensional problem converge to the solutions of the original problem as N and γ go to infinity.

Because we do not know f and $\delta f/\delta\alpha$ explicitly we use numerical methods (numerical integration of the differential equations and numerical optimization like Newton's algorithm) to compute a solution of the problem. Such a solution is said to be a near-optimal solution of the original problem.

Figure 6 from [71] shows three examples of near-optimal paths computed from this method for a mobile robot with two trailers [49].

5 Nonholonomic path planning for small-time controllable systems

Consider the following steering method for a two-driving wheel mobile robot. To go from the origin $(0,0,0)$ to some configuration (x,y,θ) the robot first

[10] This approximation restricts the family of the admissible control laws.

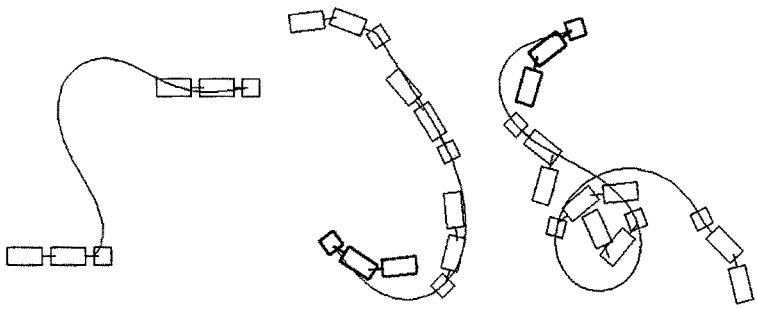


Fig. 6. Three examples of near-optimal paths.

executes a pure rotation to the configuration $(0, 0, atan\frac{y}{x})$, then it moves along a straight line segment to $(x, y, atan\frac{y}{x})$, and a final rotation steers it to the goal. This simple method accounts for local controllability: any point in any neighborhood of the origin can be reached by this sequence of three elementary paths (when $x = 0$, replace $atan\frac{y}{x}$ by $\pm\frac{\pi}{2}$). Nevertheless such a method does not account for small-time controllability. If the space is very constraint it does not hold. Think to the parking task (Figure 17): the allowed mobile robot orientations θ vary in some interval $]-\eta, \eta[$. To go from $(0, 0, 0)$ to $(0, \epsilon, 0)$ the steering method violates necessarily this constraint.

Therefore, obstacle avoidance requires steering methods accounting for small-time controllability. Such a requirement can be translated into geometric terms.

5.1 Toward steering methods accounting for small-time controllability

Let $\mathrm{d}_{\mathcal{CS}}$ be the following distance over the configuration space $\mathcal{CS}$:

$$\mathrm{d}_{\mathcal{CS}}(X^1, X^2) = \sum_{i=1}^{n} |x_i^1 - x_i^2|$$

The set of configurations X^2 such that $\mathrm{d}_{\mathcal{CS}}(X^1, X^2) < \epsilon$ is denoted by $B(X^1, \epsilon)$; this is the ball centered at X^1 with radius ϵ.

Let $\mathcal{P}$ be the set of feasible paths defined over an interval of the type $[0, T]$. A steering function is a mapping from $\mathcal{CS} \times \mathcal{CS}$ into $\mathcal{P}$:

$$(X^1, X^2) \rightarrow \mathsf{Steer}(X^1, X^2)$$

where $\mathsf{Steer}(X^1, X^2)$ is defined over the interval $[0, T]$, such that $\mathsf{Steer}(X^1, X^2)(0) = X^1$, $\mathsf{Steer}(X^1, X^2)(T) = X^2$.

Definition 2. Steer *verifies the weak topological property iff:*

$$\forall \epsilon > 0, \forall X^1 \in \mathcal{CS}, \exists \eta > 0, \forall X^2 \in \mathcal{CS}, \tag{10}$$

$$d_{\mathcal{CS}}(X^1, X^2) < \eta \;\Rightarrow\; \forall t \in [0,T],\; d_{\mathcal{CS}}(\mathsf{Steer}(X^1, X^2)(t), X^1) < \epsilon$$

By using a steering method that verifies the weak topological property, it is possible to approximate any collision-free path γ_{free}. Nevertheless, this property is not sufficient from a *computational* point of view. Indeed, it is *local*: the real number η depends on X^1. Situations as shown in Figure 7 may appear: let us consider a sequence of configurations X^i converging to the critical point X^c, and such that $\lim_{X^i \to X^c} \eta(X^i) = 0$; to be collision-free any admissible path should necessary goes through the configuration X^c. The computation of X^c may set numerical problems. To overcome this difficulty, we introduce a stronger property for the steering methods.

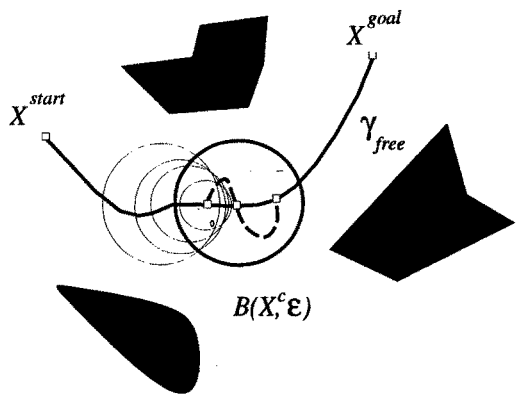


Fig. 7. Weak topological property

Definition 3. Steer *verifies the topological property iff:*

$$\forall \epsilon > 0, \exists \eta > 0, \forall (X^1, X^2) \in (\mathcal{CS})^2, \tag{11}$$

$$d_{\mathcal{CS}}(X^1, X^2) < \eta \;\Rightarrow\; \forall t \in [0,T],\; d_{\mathcal{CS}}(\mathsf{Steer}(X^1, X^2)(t), X^1) < \epsilon$$

In this definition η does not depend on any configuration (Figure 8). This is a global property that not only accounts for small-time controllability but also holds uniformly everywhere.

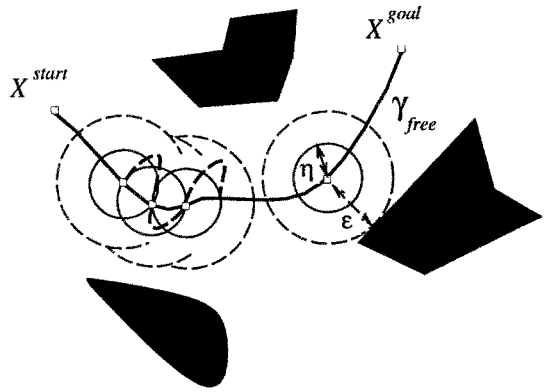


Fig. 8. Topological property

Remark 1: Proving that a given steering method verifies the topological property is not an easy task. The following sufficient condition appears in [71,74]. Let us equip $\mathcal{P}$ with a metric $d_{\mathcal{P}}$ between paths: Γ_1 and Γ_2 being two paths on $[0,1]$, we define $d_{\mathcal{P}}(\Gamma_1,\Gamma_2) = \max_{t\in[0,1]} \mathrm{d}_{\mathcal{CS}}(\Gamma_1(t),\Gamma_2(t))$.

Let us consider a steering method Steer continuous w.r.t. to the topology induced by $d_{\mathcal{P}}$. Steer is uniformly continuous on any compact set $\mathcal{K}$ included in $\mathcal{CS}^2$, i.e.,

$$\forall \epsilon > 0\ ,\ \exists \eta > 0,\ \ \forall \{(X^1,X^2),(Y^1,Y^2)\} \in \mathcal{K}$$
$$d_{\mathcal{CS}}((X^1,Y^1) < \eta \text{ and } d_{\mathcal{CS}}((X^2,Y^2) < \eta$$
$$\Longrightarrow\ d_{\mathcal{P}}(\mathsf{Steer}(X^1,X^2),\mathsf{Steer}(Y^1,Y^2)) < \epsilon$$

Choosing $X^2 = Y^1 = Y^2$ we deduce:

$$\forall \epsilon > 0\ ,\ \exists \eta > 0,\ \ \forall (X^1,X^2) \in \mathcal{CS}^2$$
$$d_{\mathcal{CS}}(X^1,X^2) < \eta \Longrightarrow d_{\mathcal{P}}(\mathsf{Steer}(X^1,X^1),\mathsf{Steer}(X^1,X^2)) < \epsilon$$

Now, let us assume that $\mathsf{Steer}(X,X) = \{X\}$ at any point X. Then:

$$\forall \epsilon > 0\ ,\ \exists \eta > 0\ \ \forall (X^1,X^2) \in \mathcal{CS}^2$$
$$d_{\mathcal{CS}}(X^1,X^2) < \eta\ \ \Longrightarrow\ \ \forall t \in [0,1],\ \ d_{\mathcal{CS}}(X^1,\mathsf{Steer}(X^1,X^2)(t)) < \epsilon$$

Therefore a sufficient condition for a steering method Steer to verify the topological property is that (1) Steer is continuous w.r.t. the topology associated with $d_{\mathcal{P}}$ and (2) the path $\mathsf{Steer}(X,X)$ is reduced to the point X. Notice that the second condition is obviously a necessary condition.

Remark 2: The first general result taking into account the necessary uniform convergence of steering methods is due to Sussmann and Liu [84]: the authors propose an algorithm providing a sequence of feasible paths that *uniformly* converge to any given path. This guarantees that one can choose a feasible path *arbitrarily close* to a given collision-free path. The method uses high frequency sinusoidal inputs. Though this approach is general, it is quite hard to implement in practice. In [87], Tilbury *et al* exploit the idea for a mobile robot with two trailers. Nevertheless, experimental results show that the approach cannot be applied in practice, mainly because the paths are highly oscillatory. Therefore this method has never been connected to a geometric planner in order to get a global planner which would take into account both environmental and kinematic constraints.

5.2 Steering methods and topological property

In this section we review the steering methods of Section 4 with respect to the topological property.

Optimal paths Let us denote by Steer_{opt} the steering method using optimal control. Steer_{opt} naturally verifies the weak topological property. Indeed the cost of the optimal paths induce a special metric in the configuration space; such a metric is said to be a nonholonomic [92], or singular [14], or Carnot-Caratheory [56], or sub-Riemannian [80] metric. By definition any optimal path with cost r does not escape the nonholonomic ball centered at the starting point with radius r. A general result states that the nonholonomic metrics induce the same topology as the "natural" metrics d_{CS}. This means that for any nonholonomic ball $B_{nh}(X,r)$ with radius r, there are two real numbers ϵ and η strictly positive such that $B(X,\eta) \subset B_{nh}(X,r) \subset B(X,\epsilon)$.

The nonholonomic distance being continuous, to get the topological property, it suffices to restrict the application of Steer_{opt} to a compact domain of the configuration space[11].

There is no general result that gives the exact shape of the nonholonomic balls; nevertheless the approximated shape of these balls is well understood (e.g., see Bellaïche–Jean–Risler's chapter).

The metric induced by the length of the shortest paths for Reeds&Shepp's car is close to a nonholonomic metric; car-like robots are the only known cases where it is possible to compute the exact shape of the balls (see Figure 1).

[11] Notice that the steering method Steer_{opt} is not necessarily continuous w.r.t. the topology induced by $d_{\mathcal{P}}$.

Sinusoidal inputs and chained form systems Let us consider the two-input chained form system (8) together with the sinusoidal inputs (9) presented in Section 4.3. We have seen that the shape of the paths depends on a_1 (Figure 4). The only constraint on the choice of a_1 is that it should be different from zero.

The steering method using such inputs is denoted by $\mathsf{Steer}_{sin}^{a_1}$. For a fixed value of a_1, $\mathsf{Steer}_{sin}^{a_1}$ does not verify the topological property. Indeed, for any configuration Z, the path $\mathsf{Steer}_{sin}^{a_1}(Z,Z)$ is not reduced to a point[12].

Therefore, the only way to build a steering method based on sinusoidal inputs and verifying the topological property is not to keep a_1 constant. One has to prove the existence of a continuous expression of $a_1(Z^1,Z^2)$ such that:

$$\lim_{Z^2 \to Z^1} a_1(Z^1,Z^2) = 0$$
$$\lim_{Z^2 \to Z^1} a_0(Z^1,Z^2,a_1(Z^1,Z^2)) = 0$$
$$\lim_{Z^2 \to Z^1} b_i(Z^1,Z^2,a_1(Z^1,Z^2)) = 0$$

The proof appears in [73]. It first states that, for a fixed value of a_1, $\mathsf{Steer}_{sin}^{a_1}$ is continuous w.r.t. to the topology induced by $d_{\mathcal{P}}$. This implies that when the final configuration Z^2 tends to the initial configuration Z^1, the path $\mathsf{Steer}_{sin}^{a_1}(Z^1,Z^2)$ tends to the path $\mathsf{Steer}_{sin}^{a_1}(Z^1,Z^1)$. Moreover, for any Z, $\mathsf{Steer}_{sin}^{a_1}(Z,Z)$ tends to $\{Z\}$ when a_1 tends to zero. Combining these two statements and restricting the application of $\mathsf{Steer}_{sin}^{a_1}$ to a compact domain $\mathcal{K}$ of $\mathcal{CS}^2$, one may conclude that:

$$\forall \epsilon > 0\ ,\ \exists A_1 > 0\ \ \forall a_1 < A_1,\ \exists \eta(a_1),\ \ \forall (Z^1,Z^2) \in \mathcal{K},$$

$$d_{CS}(Z^1,Z^2) < \eta(a_1),\ \implies\ \forall t \in [0,1],\ \ d_{CS}(Z^1, \mathsf{Steer}_{sin}^{a_1}(Z^1,Z^2)(t)) < \epsilon$$

Then, by tuning a_1, it is *a priori* possible to design a steering method Steer_{sin} based on sinusoidal inputs and verifying the topological property. It remains to define a constructive way to tune the parameter a_1. The problem is not easy. Indeed the general expression of parameters a_0 and b_i are unknown. Then we do not dispose of a unique expression of Steer_{sin}. Nevertheless, it is possible to "simulate" a steering method verifying the topological property, by switching between different $\mathsf{Steer}_{sin}^{a_1}$ according to the distance between the start and goal configurations. The principle of the construction presented in Annex consists in introducing the possibility to iteratively compute subgoals and then a sequence of subpaths that reaches the final goal without escaping a bounded domain.

[12] The first coordinate of points lying on the path is $z_1(t) = z_1 + \frac{a_1}{\omega}(1 - \cos\omega t)$.

A flatness based steering method for mobile robots with trailers We have seen in Section 4.4 that a mobile robot with two trailers (with centered hooking up system) is flat with the coordinates (x_2, y_2) of the second trailer reference point P_2 as linearizing outputs. Planning an admissible path for the system then consists in finding a sufficiently smooth planar curve $\gamma_2(s)$ for P_2. All the coordinates $(x, y, \theta, \varphi_1, \varphi_2)$ of a configuration can be geometrically deduced from $(x_2, y_2, \theta_2, \kappa_2, \frac{d}{ds_2}\kappa_2)$. Nevertheless this steering method cannot verify the topological properties. Indeed, due to the conditions on γ_2 (absence of cusp points), going from a configuration $(x_2, y_2, \theta_2, \ .\ , \ .\ )$ to some configuration $(x_2, y_2 + \epsilon, \theta_2, \ .\ , \ .\ )$ should necessarily contain a configuration $(\ .\ , \ .\ , \theta_2 \pm \frac{\pi}{2}, \ .\ , \ .\ )$.

[74] takes advantage of the flatness property of a mobile robot with one trailer to design a steering method verifying the topological property. [41] generalizes the method to a system with n trailers. Let us sketch the method for a mobile robot with two trailers.

Let us consider a configuration $X = (x, y, \theta, \varphi_1, \varphi_2)$ of the system. If Γ is an admissible path in the configuration space, we will denote by γ_2 the curve in $\mathbf{R}^2$ followed by P_2. Among all the admissible paths containing a configuration X, there exists exactly one path Γ such that $\frac{d}{ds_2}\kappa_2$ remains constant everywhere: the corresponding curve γ_2 is a clothoid[13].

Let X^{start} and X^{goal} be the initial and goal configurations respectively. Let $\gamma_{2,start}$ and $\gamma_{2,goal}$ be the associated clothoids defined on $[0, 1]$ and such that $\Gamma_{start}(0) = X^{start}$ and $\Gamma_{goal}(1) = X^{goal}$. Then any combination $\gamma(t) = \alpha(t)\gamma_{2,start}(t) + (1 - \alpha(t))\gamma_{2,goal}(t)$ is a C^3 path; it then corresponds to an admissible path for the whole system. To make this path starting at X^{start} and ending at X^{goal}, α should verify: $\alpha(0) = \dot{\alpha}(0) = \ddot{\alpha}(0) = \dddot{\alpha}\ (0) = \dot{\alpha}(1) = \ddot{\alpha}(1) = \dddot{\alpha}\ (1) = 0$ and $\alpha(1) = 1$ (indeed, the three first derivatives of γ should be the same as those of $\gamma_{2,start}$ at 0 and the same as those of $\gamma_{2,goal}$ at 1).

At this level we get a steering method (denoted by Steer^*_{flat}) that allows the mobile robot with its two trailers to reach any configuration from any other one. Nevertheless, this method does not verify the required topological property: indeed it cannot generate cusps and the steering method we want to provide should be able to do it when it is necessary.

Let X^{start} be an initial configuration and $\gamma_{2,start}$, the corresponding clothoid in $\mathbf{R}^2$. In [41] we prove that, for any $\epsilon > 0$, if we choose a configuration X within a "cone" $\mathcal{C}_{start,\epsilon}$ centered around Γ_{start} with vertex X^{start} (see Figure 9), then the path $\mathsf{Steer}^*_{flat}(X^{start}, q)$ does not escape the ball $B(X^{start}, \epsilon)$.

Moreover, if X^{start} moves within a small open set, the clothoid $\gamma_{2,start}$ is submitted to a continuous deformation: for instance a change on the coordinates

[13] If we consider only one trailer, we impose κ_1 to remain constant; in this case the trailer follows a circle with radius $\ell.\mathrm{cotan}\varphi_1$

(x_2, y_2) (respectively θ_2) of X^{start} corresponds to a translation (respectively rotation) of the clothoid $\gamma_{2,start}$. Then for a small deformation, the corresponding path in the configuration space necessarily intersects $\mathcal{C}_{start,\epsilon}$.

Let us now consider a configuration X^{goal} sufficiently close to X^{start}. The local planner Steer_{flat} then works as follows:

- If X^{goal} belongs to $\mathcal{C}_{start,\epsilon}$, $\mathsf{Steer}_{flat}(X^{start}, X^{goal}) = \mathsf{Steer}^*_{flat}(X^{start}, X^{goal})$
- otherwise, we choose a point X^{cusp} on $\gamma_{2,goal}$ within $\mathcal{C}_{start,\epsilon}$ and $\mathsf{Steer}_{flat}(X^{start}, X^{goal})$ is constituted by $\mathsf{Steer}^*_{flat}(X^{start}, X^{cusp})$ followed by the arc of the clothoid $\gamma_{2,goal}$ between X^{cusp} and X^{goal}.

With this construction $\mathsf{Steer}_{flat}(X^{start}, X^{goal})$ is guaranteed to remain within the ball $B(X^{start}, \epsilon)$ (Figure 9).

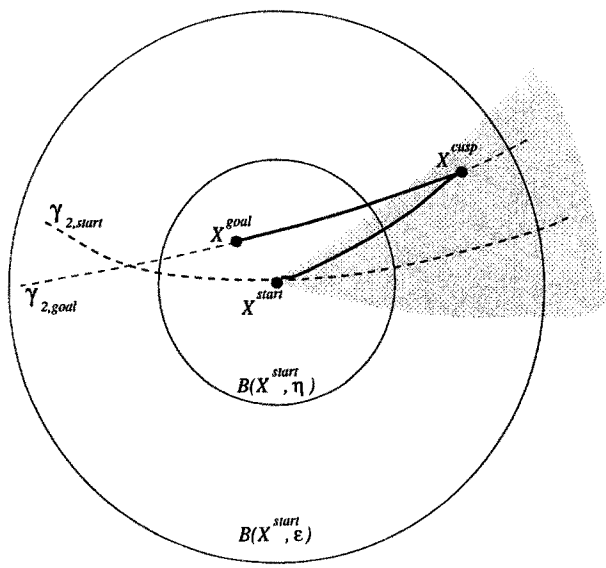


Fig. 9. Topological property of Steer_{flat}.

Figure 10 shows an example of the path generated by Steer_{flat}.

Figures 11 shows paths computed by Steer_{flat} for a mobile robot with one trailer (the curve is the path followed by the robot).

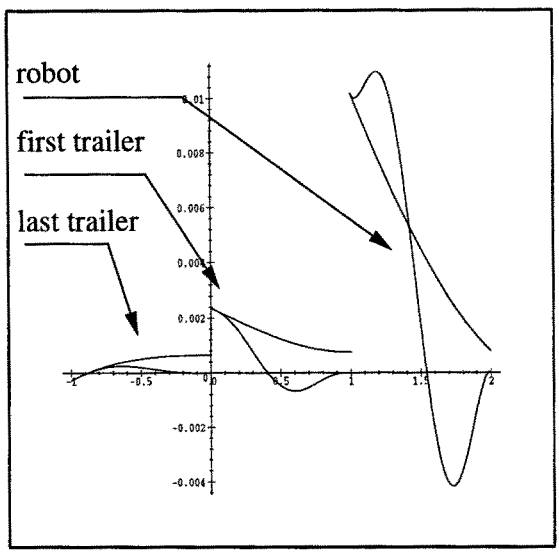


Fig. 10. An admissible path for a mobile robot with 2 trailers

5.3 Approximating holonomic paths: a two step approach

Principle Everything being in place, we may now define a first nonholonomic path planning scheme for small-time controllable systems. It consists in approximating a collision-free (holonomic) path by a sequence of collision-free admissible ones. Applying this scheme requires three main components:

- A geometric path planner that computes collision-free paths without taking into account the kinematic constraints.
- A steering method verifying the topological property.
- A geometric routine checking whether a given path is collision-free or not.

The algorithm itself is then very simple:

1. Step 1: Plan a collision-free path with the geometric path planner. If no such path exists, the algorithm stops: there is no solution.
2. Step 2: Perform subdivisions on the path until all endpoints can be linked to their neighbors by an admissible collision-free path.

Convergence and completeness: By Theorem 3.1, the convergence of Step 2 is guaranteed as soon as the steering method verifies the topological property.

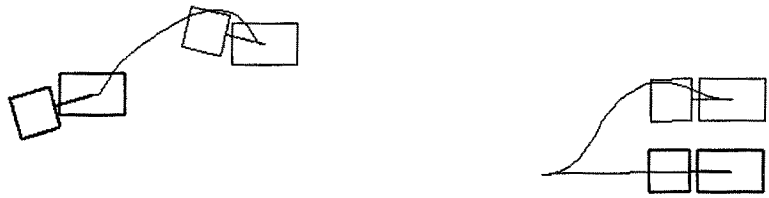


Fig. 11. Admissible paths computed by Steer_{flat} for a mobile robot with one trailer

Then the completeness of the algorithm only depends on the completeness of the geometric planner that computes a first collision-free path[14].

Geometric planner: There are no general and practical algorithm solving the classical "piano mover" problem with completeness property[15]. Numerous techniques are available to address dedicated problems [42]. Moreover new general principles appeared in the past few years. Among them one should notice the "distributed representation approach" [5] that leads to resolution-complete algorithms (such algorithms are guaranteed to find a solution when a solution exists at a given resolution when modeling the search space by a grid). Another notion is related to the behavior of probabilistic algorithms: an algorithm is said to be probabilistically complete if it includes random choices and if it is guaranteed to find a solution in finite (possibly unbounded) time when a solution exists; such algorithms cannot terminate with a negative answer on the existence of a solution. Nevertheless resolution and probabilistically complete algorithms are well understood [8] and they lead today to fast and practical motion geometric planners even for highly dimensionated systems.

Smoothing step: Step 2 provides a sequence of elementary admissible paths computed by the chosen steering method. The length of the sequence mainly depends on the clearance of the first collision-free path: the closer to the obstacles the path is, the more it should be subdivided. The sequence may be shorten in a third step by applying the steering method to link randomly chosen pairs

[14] An algorithm is complete if it is guaranteed to report a negative answer when a solution does not exist and to compute a solution otherwise.

[15] General algorithms have been provided within the framework of real algebraic geometry [69,18]; nevertheless the performance of the existing algebraic computing software and the intrinsic computational complexity of the general problem do not allow effective implementations of these algorithms.

of points lying on the first solution path. Unfortunately there is today no result insuring the convergence of this third step to any "optimal" sequence.

Several nonholonomic path planners have been designed in this way. Here is a review of the main ones.

Application to mobile robots (without trailer) The seminal ideas of the algorithm above have been introduced in [48]. This reference deals with car-like robots. When the robot is a polygon, the geometric planner is derived from the algorithm based on an analytical representation of the configuration space of a polygon moving amidst polygonal obstacles [4]. When the robot is a disk, the geometric planner works from the Voronoi diagram of the environment. In both cases, the steering method is Steer_{opt}: it consists in computing the shortest length admissible paths for a car-like robot as characterized in [65]. Due to the completeness of the geometric planners the proposed algorithms are complete. Nevertheless they are delicate to implement and fragile in practice (indeed the basic geometric routines are sensitive to numerical computations; a robust implementation could be done by using software computing with rational numbers).

Another version of this algorithm appears in [43] where the geometric planner has been replaced by a distributed representation approach; the search consists in exploring the discretized configuration space with an A^* algorithm heuristically guided by a potential function. It is then resolution-complete, less fragile than the original version, and sufficiently efficient to be integrated on real robots. Figure 12 shows an example of a solution from a software developed for the mobile robots Hilare at LAAS.

A clever idea appears in [55]. It tends to minimize the length of the shortest path sequence approximating the geometric path. It consists in computing a skeleton gathering the set of points with maximum clearance with respect to the obstacles. The key point is that the clearance is measured with the metric induced by the length of the shortest admissible paths (the so-called Reeds&Shepp metric). Then the geometric planner works by retracting the initial and goal configurations on the skeleton and by exploring it. Even if one cannot conclude to any optimality of the solution, the sequence of shortest paths provided the approximation step is shorter than a sequence approximating a path that would lie closer to the obstacles. A critical point of the approach is the computation of the metric. The distance between two configurations being invariant by translation and rotation, the authors use a lookup table storing all the distance value over a discretized compact region around the origin. This table is computed off-line once and may be used to compute the skeleton of various environments.

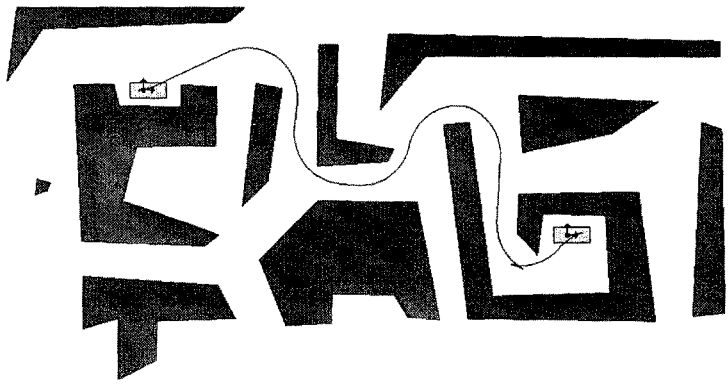


Fig. 12. A planned path for a car-like robot: the workspace is modeled by a grid of 250 x 150 pixels; the total running of the algorithm is 2 seconds on an Indy Silicon Graphics.

Recent results derived from the synthesis of the shortest paths for a car-like robot (see above and Souères–Boissonnat's Chapter) provide an analytical way to compute the shortest path distance to a polygonal obstacle for a point car-like robot [91]. This means that all the distance computations in the Reeds&Shepp metric can be done on-line. This property has been exploited to include dynamic obstacle avoidance when the robot executes its trajectory. Figure 13 from [38] shows an example of on-line updating of an admissible path when an unexpected obstacle (the black box) occurs during the execution of the motion. The various balls covering the path in the figure are the projection onto $\mathbf{R}^2$ of the maximal collision-free Reeds&Shepp balls covering the path in the configuration space. Up to now, the distance function are known for a point robot; its extension to a polygonal robot has to be done.

The case of mobile robots with trailers The case of mobile robots with n trailers has been solved by using RPP as geometric planner and the three steering methods Steer_{opt}, Steer_{sin} (for $n = 1$ and $n = 2$) and Steer_{flat} (for $n = 1$) [71].

To compute a collision-free path we use the algorithm RPP, the random path planner presented in [5]. We consider $n+1$ control points: two are located on the robot and one is located on each trailer. The start configuration and the goal being given, a potential field is computed for each control point in the workspace[16]; the $n+1$ potential fields are then combined to create a potential field in the configuration space; the search consists in following the gradient of

[16] The potential field are computed from a bitmap representation of the workspace.

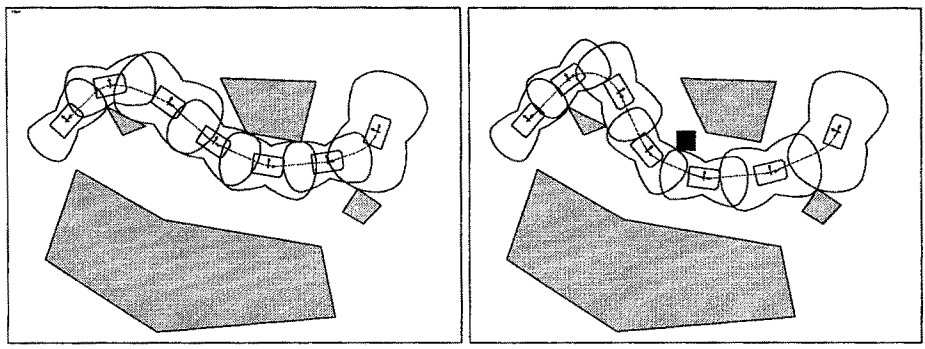


Fig. 13. A planned path is updated in real-time when an unexpected moving obstacle occurs.

the potential; when it stops at some local minimum, the algorithm generates a random path and follows again the gradient until the goal is reached. The algorithm is probabilistically complete.

From the various experiments reported in [71], it appears that the the algorithm based on Steer_{sin} is much faster than the algorithm based on Steer_{opt} (in terms of computation time) for a mobile robot with one or two trailers. For a mobile robot with one trailer the computation time are roughly the same when using Steer_{flat} and Steer_{sin}; nevertheless the smoothness of the final path is better with Steer_{flat} than with Steer_{sin}.

In the examples on Figures 14 and 15 the workspace is modeled by a grid of 600×470 pixels.

5.4 Probabilistic approaches

Svestka–Overmars' chapter reviews recent results provided by applying a new general paradigm in motion planning. This is a probabilistic approach consisting in two phases:

- In a first learning phase an incremental roadmap is built by randomly choosing collision-free configurations and by linking them with admissible paths. Admissible paths are computed with a (not necessarily complete) local path planner.
- In the query phase, paths are to be found between some given start and goal configurations. The local path planner is used to connect the configurations to some nodes of the roadmap. If this succeeds, a graph search is performed.

As for the approach using a holonomic path approximation, the algorithm includes a last step consisting in smoothing the computed solution.

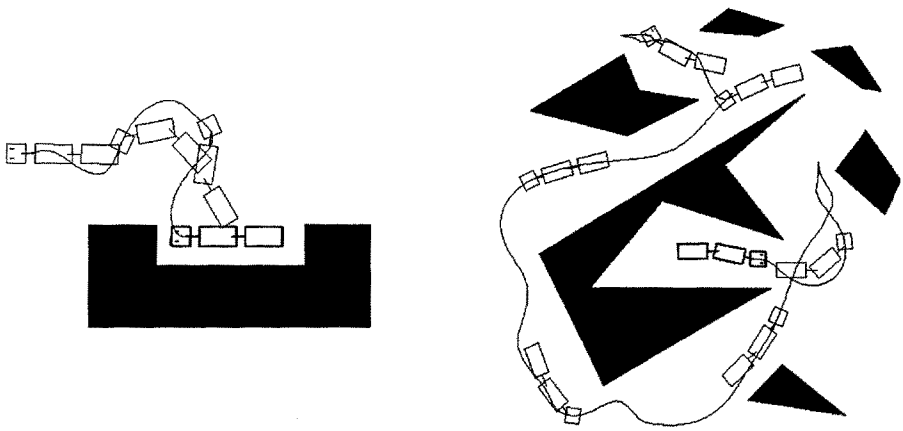


Fig. 14. Solutions using Steer_{sin}: the total computation time is 30 seconds (left) and 114 seconds (right) on a Sun-Sparc-20.

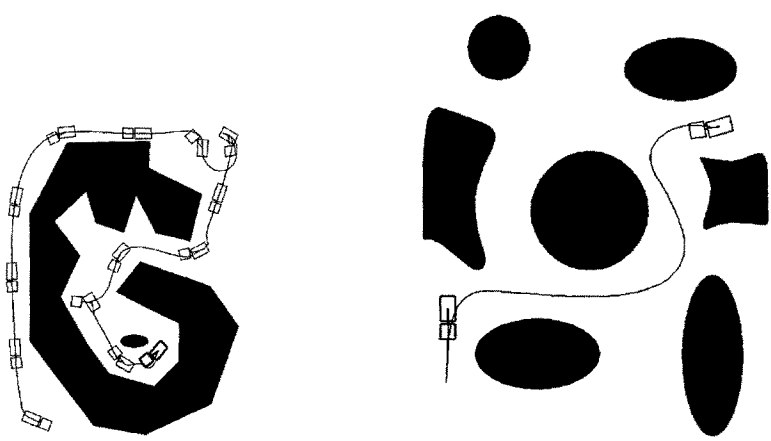


Fig. 15. Solutions using Steer_{flat}: the total computation time is 21 seconds (left) and 6 seconds (right) on a Sun-Sparc-20.

Such a scheme applies for nonholonomic systems as soon as the local path planner is a steering method verifying the topological property. The algorithm is probabilistically complete. It has been applied to mobile robots with trailers on the basis of Steer_{sin} [70]. An analysis of the approach together with practical results are overviewed in Svestka–Overmars' chapter.

5.5 An approach using optimization techniques

At the same time, a slightly different approach has been proposed by Bessière *et al* [12]. Its principle consists in exploring the free space from the initial configuration along admissible paths by spreading landmarks, each being as far as possible from one another. In parallel, a local path planner checks if the target may be reached from each new landmark. Both phases are solved by using optimisation techniques (e.g., genetic algorithms). This general paradigm has been applied to nonholonomic mobile robots in [3] by using the Steer_{sin} as local path planner. Because Steer_{sin} verifies the topological property the algorithm may be proved to be complete as soon as the convergence the optimization routines is guaranteed.

5.6 A multi-level approach

It remains that the computational cost of the nonholonomic path planners increases with the dimension of the systems. Facing the intrinsic complexity of the problem for practical applications requires a good understanding of the kinematic structures of the systems as well as a good experience in evaluating the performance of a given planning scheme. [70] presents a multi-level nonholonomic path planner.

Let us illustrate the idea from a car-like robot pulling two trailers: from the collision avoidance point of view the system is of dimension five (three parameters for the car and one parameter for each trailer); from the control point of view the direction of the front wheels of the car is taken into account: the system is then six-dimensionated.

The underlying idea consists in introducing the nonholonomic constraints of the bodies *iteratively*. In a first step one plans a "semi-holonomic" path feasible by the car, but not necessarily by the trailers (i.e. at this step the trailers are assumed to be holonomic). Then the nonholonomic constraint due to the first trailer is introduced: this step consists in searching a path feasible by both the car and the first trailer. Finally, all the kinematic constraints are taken into account.

Each step should benefit from the path computed by the previous one, via a specific nonholonomic motion planner. In [70], the first semi-holonomic path is computed with a probabilistic approach that considers only the kinematic

constraints of the car. Then a probabilistic search using Steer_{sin} is applied within a tube surrounding the path; it provides a second semi-holonomic path that takes into account the first three kinematic constraints. Finally the second path is approximated via Steer_{sin} accounting for all the constraints. The global algorithm is then based on a combination of the holonomic path approximation scheme and the probabilistic one.

Three examples of solutions provided by the algorithm appear in Figure 16: the left column shows the first "semi-holonomic" paths (the two trailers "slide"); the right column shows the corresponding final paths. The total time to compute the solutions ranges from less than one minute for the first example to around three minutes for the third one, on a 136 MIPS workstation.

5.7 On the computational complexity of nonholonomic path planning

Evaluating the computational complexity of the approaches introduced above is a difficult task. More generally, the complexity of the nonholonomic path planning problem is an open problem.

For small-time controllable systems, we have seen that the existence of a solution is characterized by the existence of any collision-free path for the associated holonomic system. The complexity of deciding whether a solution exists is then equivalent to the complexity of the classical piano mover problem (see [42] for an overview). The complexity for other systems (e.g., with drift) is an open problem.

In this section we give an account of results providing lower bounds on the complexity of nonholonomic paths for small-time controllable mobile robots. By reference to the approximation scheme, we may define the complexity of a collision-free nonholonomic path by the length of the sequence of admissible paths approximating a holonomic one. This definition depends a priori on the steering method used to approximate a holonomic path. A more intrinsic definition consists in considering the approximation scheme that uses Steer_{opt}. Indeed the cost of the optimal paths induces a (nonholonomic) metric in the configuration space. A possible definition of the complexity of a path is the minimum number of balls computed with the nonholonomic metric and covering the path. For instance the complexity of the paths appearing in Figure 13 is 7 in both cases. This definition allows to link the complexity of nonholonomic path planning with the clearance of the free-space.

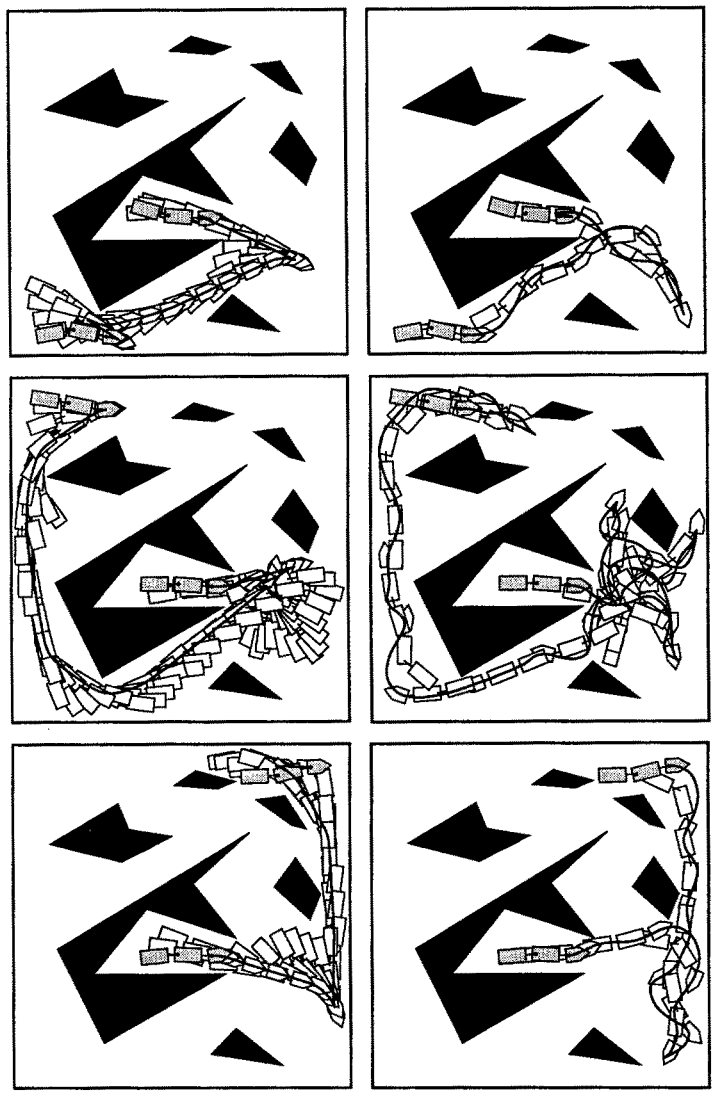


Fig. 16. Examples of solutions computed by the multi-level approach (the left column shows the first "semi-holonomic" paths)

Let us consider the classical parking task problem illustrated in Figure 17 for a car-like robot. The solutions have been computed by the algorithm presented in Section 5.3. The steering method to approximate the holonomic path is Steer_{opt} which computes Reeds&Shepp's shortest paths. The length of the shortest paths induces a metric d_{RS} in configuration space. The shape of the balls computed with this metric appears in Figure 1 (top). Let us consider a configuration $X = (x, y, \theta)$ near the origin O. It has been proved in [48] that:

$$\frac{1}{3}(|x| + |y|^{\frac{1}{2}} + |\theta|) \leq d_{RS}(O, X) \leq 12(|x| + |y|^{\frac{1}{2}} + |\theta|)$$

As a consequence, the number of balls required to cover the "corridor" where the car has to be parked varies as ϵ^{-2} with ϵ being the width of the corridor. Moreover each elementary shortest path providing a motion in the direction of the wheel axis requires exactly two cusps. Then the number of maneuvers to park a car is in $\Omega(\epsilon^{-2})$.

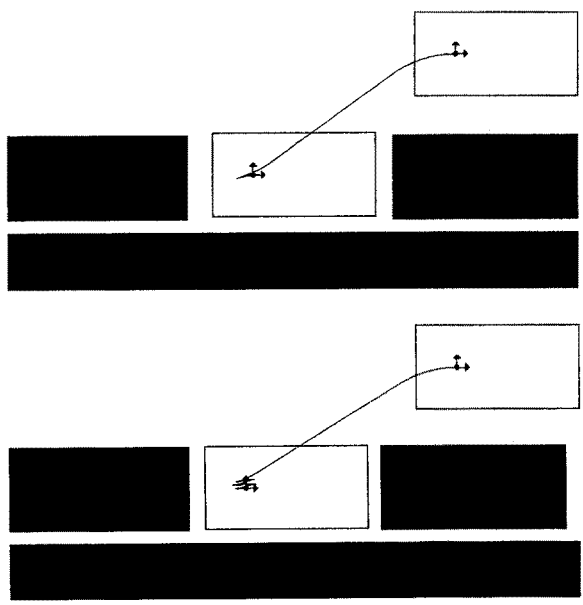


Fig. 17. The number of maneuvers varies as the inverse of the square of the free space.

Such a reasoning may be generalized to small-time controllable systems. Let us consider a control system defined by a set of vector fields; let us assume

that the tangent space at every point can be spanned by a finite family of these vector fields together with their Lie brackets (i.e., the system verifies the LARC at every point). The minimal length of the Lie bracket required to span the tangent space at a point is said to be the degree of nonholonomy of the system at this point.

The cost of the optimal paths induces a metric in the configuration space of the system. A ball of radius r corresponding to this metric is the set of all the points in the configuration space reachable by a path of cost lesser than r. The balls grow faster in the directions given by the vector fields directly controlled than in the directions defined by the Lie brackets of these vector fields. A powerful result from sub-Riemannian geometry shows that the growing law depends on the degree of bracketing (see [9,29,92,56] or Bellaïche–Jean–Risler's chapter): when r is small enough, the ball grows as r in the directions directly controlled; it grows as r^d in the directions spanned by Lie brackets of length d.

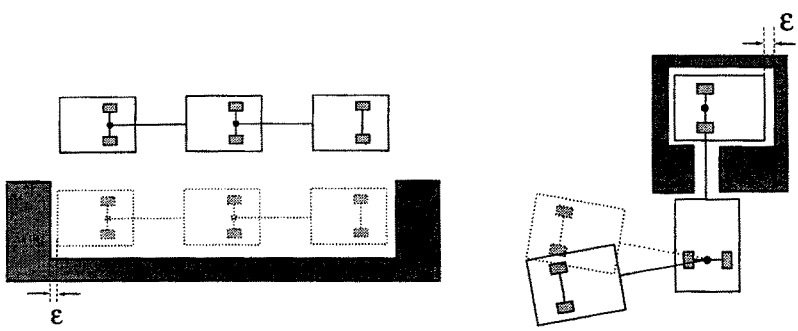


Fig. 18. The complexity of admissible paths for a mobile robot with n trailers are respectively $\Omega(\epsilon^{-n-2})$ (case on the left side) and $\Omega(\epsilon^{-Fib(n+3)})$ (case on the right side).

Figure 18 illustrates this complexity modeling on a mobile robot with two trailers. We have seen in Section 2.3 that the degree of nonholonomy of this system is 4 when $\varphi_1 \neq \frac{\pi}{2}$ (regular points) and 5 everywhere else. This means that the complexity of the parking task is in $\Omega(\epsilon^{-4})$ while the complexity of the exotic example on the right side (the mobile robot can not escape from the room ...) is in $\Omega(\epsilon^{-5})$. These worst case examples may be generalized to an arbitrary number of trailers: the degree of nonholonomy for a mobile robot with n trailers has been proved to be $n+2$ at regular points and $Fib(n+3)$ when all the relative angles of the trailers are $\frac{\pi}{2}$ [54,36] ($Fib(n+3)$ is the $(n+3)$th number of the famous sequence of Fibonacci defined by $Fib(i+2) = Fib(i+1) + Fib(i)$, i.e., 1, 1, 2, 3, 5, 8, 13 ...). This means that the complexity of the problems

appearing in Figure 18 and generalized to n trailers are respectively $\Omega(\epsilon^{-n-2})$ (simply exponential in n) and $\Omega(\epsilon^{-Fib(n+3)})$ (doubly exponential in n).

6 Other approaches, other systems

This section overviews other works related to nonholonomic path planning for mobile robots. They deal either with direct approaches based on dynamic programming techniques, or with specific systems.

Combining discrete configuration space and piece-wise constant inputs: Barraquand and Latombe propose in [6,7] a direct approach to nonholonomic path planning. It applies to car-like robots with trailers. The model of the car corresponds to the control system (4) introduced in Section 2.2. Four input types are chosen in $\{-1,1\} \times \{\zeta_{min}, \zeta_{max}\}$; they correspond to backward or forward motions with an extremal steering angle. The admissible paths are generated by a sequence of these constant inputs, each of them being applied over a fixed interval of time δt. Starting from the initial configuration the search generates a tree: the successors of a given configuration X are obtained by setting the input to one of the four values and integrating the differential system over δt. The configuration space is discretized into an array of cells of equal size (i.e. hyperparallelepipeds). A successor X' of a configuration X is inserted in the search tree if and only if the computed path from X to X' is collision-free and X' does not belong to a cell containing an already generated configuration. The algorithm stops when it generates a configuration belonging to the same cell as the goal (i.e., it does not necessarily reach the goal exactly).

The algorithm is proved to be asymptotically complete w.r.t. to both δt and the size of the cells. As a brute force method, it remains quite time-consuming in practice. Its main interest is that the search is based on Dijkstra's algorithm which allows to take into account optimality criteria such that the path length or the number of reversals. Asymptotical optimality to generate the minimum of reversals is proved for the car-like robot alone.

Progressive constraints: In [23] Ferbach combines the two step approach presented in Section 5.3 and a so-called variational approach. It applies for small-time controllable system. First, a collision-free path is generated. Then the nonholonomic constraints are introduced progressively. At each iteration, a path is generated from the previous one to satisfy more severe nonholonomic constraints. The search explores the neighborhood of the current path according to a dynamic programming procedure. The progressiveness of the search is obtained by taking random tangent vectors chosen in neighborhoods of the admissible ones and by making these neighborhoods decreasing to the set of admissible tangent vectors.

The method is neither complete nor asymptotically complete. Completeness would require back-tracking that would be expensive. Nevertheless simulations have been performed with success for a mobile robot with three trailers and for two tractor-trailer robots sharing the same environment.

Car-like robots moving forward: After the pioneering work of Dubins who characterized the shortest paths for a particle moving with bounded curvature [22], attempts have been done to attack the path planning for car-like robots moving only forward. Except some algorithms that do not verify any general completeness properties (e.g., [45,89,94]), they are only few results addressing the general problem. All of them assume that the robot is reduced to a point. In [27], Fortune and Wilfong propose an algorithm running in exponential time and space to decide if a path exists; the algorithm does not generate the solution. Jacobs and Canny's algorithm [34] is a provably good approximation algorithm that generates a sequence of elementary feasible paths linking configurations in contact with the obstacles. According to the resolution of a contact space discretization, the algorithm is proved to compute a path which is as close as possible to the minimal length path. More recent results solve the problem exactly when the obstacles are bounded by curves corresponding to admissible paths (i.e., the so-called moderate obstacles) [2,13].

Nonholonomic path planning for Dubins' car then remains a difficult and open problem[17].

Multiple mobile robots: Nonholonomic path planning for the coordination of multiple mobiles robots sharing the same environment has been addressed along two main axis: centralized and decentralized approaches[18].

In the centralized approaches the search is performed within the Cartesian product of the configuration spaces of all the robots. While the problem is PSPACE-complete [32], recent results by Svestka and Overmars show that it is possible to design planners which are efficient in practice (until five mobile robots) while being probabilistically complete [85]: the underlying idea of the algorithm is to compute a probabilistic roadmap constituted by elementary (nonholonomic) paths admissible for all the robots considered separately; then the coordination of the robots is performed by exploring the Cartesian product of the roadmaps. The more dense is the initial roadmap, the higher is the probability to find a solution in very cluttered environments.

In [1], Alami reports experiments involving ten mobile robots on the basis of a fully decentralized approach: each robot builds and executes its own plan by

[17] Notice that Barraquand and Latombe's algorithm [6] may be applied to provide an approximated solution of the problem.

[18] We refer the reader to Svestka–Overmars' chapter for a more detailed overview on this topic.

merging it into a set of already coordinated plans involving other robots. In such a context, planning is performed in parallel with plan execution. At any time, robots exchange information about their current state and their current paths. Geometric computations provide the required synchronization along the paths. If the approach is not complete (as a decentralized schemes), it is sufficiently well grounded to detect deadlocks. Such deadlocks usually involve only few robots among the fleet; then they may be overcome by applying a centralized approach locally.

7 Conclusions

The algorithmic tools presented in this chapter show that the research in motion planning for mobile robots reaches today a level of maturity that allows their transfer on real platforms facing difficult motion tasks.

Numerous challenging questions remain open at a formal level. First of all, there is no nonholonomic path planner working for any small-time controllable system. The case of the mobile robot with trailers shown in Figure 2 (right) is the simplest canonical example which can conduce new developments. A second issue is path planning for controllable and not small-time controllable systems; Dubins' car appears as another canonical example illustrating the difficulty of the research on nonhonomic systems. Souères–Boissonnat's chapter emphasizes on recent results dealing with the computation of optimal controls for car-like robots; it appears that extending these tools to simple systems like two-driving wheel mobile robots is today out of reach.

Perhaps the most exciting issues come from practical applications. The motion of the robot should be performed in the physical world. The gap between the world modeling and the real world is critical. Usually, path planning assumes a two-steps approach consisting in planning a path and then executing it via feedback control. This assumption holds under the condition that the geometric model of the environment is accurate and that the robot's Cartesian coordinates are directly and exactly measured. Designing a control law that executes a planed path defined in a robot centered frame may be sufficient in manufacturing applications; it is not when dealing with applications such as mobile robot outdoor navigation for instance. In practice, the geometric model of the world and the localisation of the robot should be often performed through the use of embarked extereoceptive sensors (ultrasonic proximeters, infrared or laser range finder, laser or video cameras ...).

Uncertainties and sensor-based motions are certainly the two main keywords to be considered to reach the ultimate objectives of the motion planning. Addressing these issues requires to revisit the motion planning problem statement: the problem is to plan not a robot-centered path but a sequence of

sensor-based motions that guaranty the convergence to the goal. The solution is no more given by a simple search in the collision-free configuration space. This way is explored in manufacturing applications for several years; it is difficult in mobile robotics where nonholonomy adds another difficulty degree.

Annex: Sinusoidal inputs and obstacle avoidance (comments on the tuning of a_1)

As we have seen in Section 5.2, we do not dispose of a unique expression of Steer_{sin} verifying the topological property. In this annex we show that it is possible to switch between different $\text{Steer}_{sin}^{a_1}$ to integrate such a steering method within a general nonholonomic path planning scheme.

Let us consider the two input chained form system (8) introduced in Section 4.3:

$$\begin{cases} \dot{z}_1 = v_1 \\ \dot{z}_2 = v_2 \\ \dot{z}_3 = z_2.v_1 \\ \vdots \quad = \vdots \\ \dot{z}_n = z_{n-1}.v_1 \end{cases}$$

$\text{Steer}_{sin}^{a_1}$ is defined by:

$$\begin{cases} v_1(t) = a_0 + a_1 \sin \omega t \\ v_2(t) = b_0 + b_1 \cos \omega t + b_2 \cos 2\omega t + \ldots b_{n-2} \cos(n-2)\omega t \end{cases}$$

We have proved that for a given a_1 small enough, the maximal gap between Z^{start} and the path $\text{Steer}_{sin}^{a_1}(Z^{start}, Z^{goal})$ decreases when Z^{goal} tends to Z^{start}. But this gap do not tends to zero. In other words, for a fixed value of a_1, trying to reach closer configurations on the geometric path decreases the risk of collision but does not eliminate it. Moreover to tend this gap to zero we have also to decrease $|a_1|$. But these two decreasings are not independent. Indeed, by changing the value of a_1 we change the steering method $\text{Steer}_{sin}^{a_1}$ and so we change the family of the paths. For a given couple of extremal configurations, a decreasing of a_1 increases in most of the cases the extremal gap between the start point and the path. In other words, in order to reduce the risk of collision we have to choose close goal configurations but we also have to reduce a_1. Which in turn increase again the clearance between the path and the start point. So we have again to bring the goal closer ... If the decreasing of $|a_1|$ is too fast with respect to the one of the distance between the start configuration and the current goal, the approximation algorithm will not converge.

A strategy for tuning these two decreasings can be integrated in the approximation algorithm (Section 5.3) while respecting its completeness. The following approach has been implemented; it is described with details in [72,71]. It is based on a lemma giving an account of the distance between a path generated by $\text{Steer}_{sin}^{a_1}$ and its starting point Z^0. Let us denote $z_i(t) - z_i^0$ by $\delta_i(t)$.

Lemma 7.1. *For any path computed by* $\text{Steer}_{sin}^{a_1}$, *for any* $t \in [0,T]$:

$$\begin{array}{ll} |\delta_1(t)| & \leq |a_0 T| + |a_1 T| = \Delta_1 \\ |\delta_2(t)| & \leq \sum |b_i T| = \Delta_2 \\ |\delta_{k+1}(t)| & \leq |z_k^0|\Delta_1 + \ldots + |z_3^0|\Delta_1^{k-2} + (|z_2^0| + \Delta_2)\Delta_1^{k-1} \quad with k > 2 \end{array} \tag{12}$$

Proof: By definition $\dot{\delta}_1(t) = a_0 + a_1 \sin \omega t$. Then:

$$|\delta_1(t)| \leq \int_0^t |\dot{\delta}_1(\tau)| d\tau \leq \int_0^t (|a_0| + |a_1|)\ d\tau \leq |a_0 T| + |a_1 T|$$

By setting $\Delta_1 = |a_0 T| + |a_1 T|$ we have the intermediate result that for all t, $\int_0^t |\dot{\delta}_1(\tau)| d\tau \leq \Delta_1$. The same reasoning holds to prove that $|\delta_2(t)| \leq \sum |b_i T|$.

Now, for any $k > 2$:

$$\delta_{k+1}(t) = \int_0^t z_k(\tau)\dot{z}_1(\tau)\ d\tau = \int_0^t \delta_k(\tau)\dot{z}_1(\tau)\ d\tau + z_k^0 \int_0^t \dot{z}_1(\tau)\ d\tau$$

An upper bound Δ_k on $|\delta_k(t)|$ being given, we get:

$$|\delta_{k+1}(t)| \leq \Delta_k \int_0^t |\dot{z}_1(\tau)|\ d\tau + |z_k^0| \int_0^t |\dot{z}_1(\tau)|\ d\tau \leq (\Delta_k + |z_k^0|)\Delta_1$$

Then

$$\Delta_{k+1} \leq (\Delta_k + |z_k^0|)\Delta_1$$

And by recurrence:

$$|\delta_{k+1}(t)| \leq |z_k^0|\Delta_1 + \ldots + |z_3^0|\Delta_1^{k-2} + (|z_2^0| + \Delta_2)\Delta_1^{k-1} \quad \square$$

Given a start configuration Z^{start}, we first fix the value of a_1 and two other parameters Δ_1^{min} and Δ_2^{min} to some arbitrary values (see [71] for details on initialization). Then we choose a goal configuration on the straight line segment $[Z^{start}, Z^{goal}]$ (or on any collision-free path linking Z^{start} and Z^{goal}]) closer and closer to Z^{start}. This operation decreases the parameters $a_0, b_0, \ldots, b_n$ so it decreases Δ_1 and Δ_2 (the detailed proof of this statement appears in [71,74]). We continue to bring the goal closer to the initial configuration until a collision-free path is found or until $\Delta_1 \leq \Delta_1^{min}$ and $\Delta_2 \leq \Delta_2^{min}$. In the second case, we substitute a_1, Δ_1^{min} and Δ_2^{min} respectively by $k.a_1$, $k.\Delta_1^{min}$ and $k.\Delta_2^{min}$, with $k < 1$ and we start the above operations again. The new starting path may or may not go further away from Z^{start} than the previous one but in any case, from equations (12) we have the guarantee that following this strategy, the computed path will lie closer and closer to Z^{start}. We have then the guarantee of finding a collision-free path.

References

1. R. Alami, "Multi-robot cooperation based on a distributed and incremental plan merging paradigm," *Algorithms for Robotic Motion and Manipulation, WAFR'96*, J.P. Laumond and M. Overmars Eds, A.K. Peters, 1997.
2. P.K. Agarwal, P. Raghavan and H. Tamaki, "Motion planning for a steering-constrained robot through moderate obstacles," *ACM Symp. on Computational Geometry*, 1995.
3. J.M. Ahuactzin, " Le Fil d'Ariane: une méthode de planification générale. Application à la planification automatique de trajectoires," PhD Thesis, INP, Grenoble, 1994.
4. F. Avnaim, J. Boissonnat and B. Faverjon, "A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles," *IEEE Int. Conf. on Robotics and Automation*, pp. 1656–1661, Philadelphia, 1988.
5. J. Barraquand and J.C. Latombe, "Robot motion planning: a distributed representation approach," *International Journal of Robotics Research*, 1991.
6. J. Barraquand and J.-C. Latombe, "On non-holonomic mobile robots and optimal maneuvering," *Revue d'Intelligence Artificielle*, Vol. 3 (2), pp. 77–103, 1989.
7. J. Barraquand and J.C. Latombe, "Nonholonomic multibody mobile robots: controllability and motion planning in the presence of obstacles," *Algorithmica*, Springer Verlag, Vol. 10, pp. 121–155, 1993.
8. J. Barraquand, L. Kavraki, J.C. Latombe, T.Y. Li, R. Motvani and P. Raghavan, "A random sampling scheme for path planning," *Robotics Research, the Seventh International Symposium*, G. Giralt and G. Hirzinger Eds, Springer Verlag, 1996.
9. A. Bellaïche, J.P. Laumond and P. Jacobs, "Controllability of car-like robots and complexity of the motion planning problem," *Int. Symposium on Intelligent Robotics*, pp. 322–337, Bangalore, 1991.
10. A. Bellaïche, J.P. Laumond and J.J. Risler, "Nilpotent infinitesimal approximations to a control Lie algebra," *IFAC Nonlinear Control Systems Design Symposium*, pp. 174–181, Bordeaux, 1992.
11. A. Bellaïche, J.P. Laumond and M. Chyba, "Canonical nilpotent approximation of control systems: application to nonholonomic motion planning," *32nd IEEE Conf. on Decision and Control*, San Antonio, 1993.
12. P. Bessière, J.M. Ahuactzin, E. Talbi and E. Mazer, "The Ariadne's clew algorithm: global planning with local methods," *Algorithmic Foundations of Robotics*, K. Goldberg *et al* Eds, A.K. Peters, 1995.
13. J.D. Boissonnat and S. Lazard, "A polynomial-time algorithm for computing a shortest path of bounded curvature amidst moderate obstacle," *ACM Symp. on Computational Geometry*, 1996.
14. R.W. Brockett, "Control theory and singular Riemannian geometry," *New Directions in Applied Mathematics*, Springer-Verlag, 1981.
15. X.N. Bui, P. Souères, J.D. Boissonnat and J.P. Laumond, "The shortest path synthesis for nonholonomic robots moving fordwards," *IEEE Int. Conf. on Robotics and Automation*, Atlanta, 1994.
16. L. Bushnell, D. Tilbury and S. Sastry, "Steering three-input nonholonomic systems: the fire-truck example," *International Journal of Robotics Research*, Vol. 14 (4), pp. 366–381, 1995.

17. G. Campion, G. Bastin and B. d'Andréa-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE Trans. on Robotics and Automation*, Vol. 12 (1), 1996.
18. J. Canny, *The Complexity of Robot Motion Planning*, MIT Press, 1988.
19. J. Canny, A. Rege and J. Reif, "An exact algorithm for kinodynamic planning in the plane," *Discrete and Computational Geometry*, Vol. 6, pp. 461–484, 1991.
20. B. Donald, P. Xavier, J. Canny and J. Reif, "Kinodynamic motion planning," *J. of the ACM*, Vol. 40, pp. 1048–1066, 1993.
21. B. Donald and P. Xavier, "Provably good approximation algorithms for optimal kinodynamic planning: robots with decoupled dynamic bounds," *Algorithmica*, Vol. 14, pp. 443–479, 1995.
22. L. E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, Vol. 79, pp. 497–516, 1957.
23. P. Ferbach, "A method of progressive constraints for nonholonomic motion planning," *IEEE Int. Conf. on Robotics and Automation*, pp. 2929–2955, Minneapolis, 1996.
24. C. Fernandes, L. Gurvits and Z.X. Li, "A variational approach to optimal nonholonomic motion planning," *IEEE Int. Conf. on Robotics and Automation*, pp. 680–685, Sacramento, 1991.
25. S. Fleury, P. Souères, J.P. Laumond and R. Chatila, "Primitives for smoothing mobile robot trajectories," *IEEE Transactions on Robotics and Automation*, Vol. 11 (3), pp. 441–448, 1995.
26. M. Fliess, J. Lévine, P. Martin and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *Int. Journal of Control*, Vol. 61 (6), pp. 1327–1361, 1995.
27. S.J. Fortune and G.T. Wilfong, "Planning constrained motions," *ACM STOCS*, pp. 445–459, Chicago, 1988.
28. T. Fraichard, "Dynamic trajectory planning with dynamic constraints: a 'state-time space' approach," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1393–1400, Yokohama, 1993.
29. V. Y. Gershkovich, "Two-sided estimates of metrics generated by absolutely nonholonomic distributions on Riemannian manifolds," *Soviet Math. Dokl.*, Vol. 30 (2), 1984.
30. G. Giralt, R. Sobek and R. Chatila, "A multi-level planning and navigation system for a mobile robot: a first approach to Hilare," *6th Int. Joint Conf. on Artificial Intelligence*, pp. 335–337, Tokyo, 1979.
31. H. Hermes, A. Lundell and D. Sullivan, "Nilpotent bases for distributions and control systems," *J. of Differential Equations*, Vol. 55, pp. 385–400, 1984.
32. J. Hopcroft, J.T. Schwartz and M. Sharir, "On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the warehouseman's problem," *International Journal of Robotics Research*, Vol. 3, pp. 76–88, 1984.
33. G. Jacob, "Lyndon discretization and exact motion planning," *European Control Conference*, pp. 1507–1512, Grenoble, 1991.
34. P. Jacobs and J. Canny, " Planning smooth paths for mobile robots," *IEEE Int. Conf. on Robotics and Automation*, Scottsdale, 1989.

35. P. Jacobs, A. Rege and J.P. Laumond, "Non-holonomic motion planning for Hilare-like robots," *Int. Symposium on Intelligent Robotics*, pp. 338–347, Bangalore, 1991.
36. F. Jean, "The car with N trailers: characterization of the singular configurations," *ESAIM: COCV*, http://www.emath.fr/cocv/, Vol. 1, pp. 241–266, 1996.
37. Y. Kanayama and N. Miyake, "Trajectory generation for mobile robots," *Robotics Research*, Vol. 3, MIT Press, pp. 333–340, 1986.
38. M. Khatib, H. Jaouni, R. Chatila and J.P. Laumond, " Dynamic path modification for car-like nonholonomic mobile robots," *IEEE Int. Conf. on Robotics and Automation*, Albuquerque, 1997.
39. G. Lafferriere and H.J. Sussmann, "A differential geometric approach to motion planning," *Nonholonomic Motion Planning*, Zexiang Li and J.F. Canny Eds, The Kluwer International Series in Engineering and Computer Science 192, 1992.
40. F. Lamiraux and J.P. Laumond, "From paths to trajectories for multi-body mobile robots," *Int. Symposium on Experimental Robotics*, Lecture Notes on Control and Information Science, Springer-Verlag, (to appear) 1997.
41. F. Lamiraux and J.P. Laumond, "Flatness and small-time controllability of multibody mobile robots: applications to motion planning," *European Conference on Control*, Brussels, 1997.
42. J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.
43. J.C. Latombe, "A Fast Path Planner for a Car-Like Indoor Mobile Robot," *Ninth National Conference on Artificial Intelligence, AAAI*, pp. 659–665, Anaheim, 1991.
44. J.P. Laumond, "Feasible trajectories for mobile robots with kinematic and environment constraints," *Intelligent Autonomous Systems*, L.O. Hertzberger, F.C.A. Groen Edts, North-Holland, pp. 346–354, 1987.
45. J.P. Laumond, "Finding collision-free smooth trajectories for a non-holonomic mobile robot," *10th International Joint Conference on Artificial Intelligence*, pp. 1120–1123, Milano, 1987.
46. J.P. Laumond, " Singularities and topological aspects in nonholonomic motion planning," *Nonholonomic Motion Planning*, Zexiang Li and J.F. Canny Eds, The Kluwer International Series in Engineering and Computer Science 192, 1992.
47. J.P. Laumond, "Controllability of a Multibody Mobile Robot," *IEEE Transactions Robotics and Automation*, pp. 755–763, Vol. 9 (6), 1993.
48. J.P. Laumond, P. Jacobs, M. Taïx and R. Murray, "A motion planner for nonholonomic mobile robot," *IEEE Trans. on Robotics and Automation*, Vol. 10, 1994.
49. J.P. Laumond, S. Sekhavat and M. Vaisset, "Collision-free motion planning for a nonholonomic mobile robot with trailers," *4th IFAC Symp. on Robot Control*, pp. 171–177, Capri, 1994.
50. J.P. Laumond and J.J. Risler, "Nonholonomic systems: controllability and complexity," *Theoretical Computer Science*, Vol. 157, pp. 101–114, 1996.
51. J.P. Laumond and P. Souères, "Metric induced by the shortest paths for a car-like robot," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Yokoama, 1993.
52. Z. Li and J.F. Canny Eds, *Nonholonomic Motion Planning*, Kluwer Academic Publishers, 1992.

53. T. Lozano-Pérez, "Spatial planning: a configuration space approach," *IEEE Trans. Computers*, Vol. 32 (2), 1983.
54. F. Luca and J.J. Risler, "The maximum of the degree of nonholonomy for the car with n trailers," in *IFAC Symp. on Robot Control*, pp. 165–170, Capri, 1994.
55. B. Mirtich and J. Canny, "Using skeletons for nonholonomic path planning among obstacles," *IEEE Int. Conf. on Robotics and Automation*, Nice, 1992.
56. J. Mitchell, "On Carnot-Carathéodory metrics," *J. Differential Geometry*, Vol. 21, pp. 35-45, 1985.
57. S. Monaco and D. Normand-Cyrot, "An introduction to motion planning under multirate digital control," *IEEE Int. Conf. on Decision and Control*, pp. 1780–1785, Tucson, 1992.
58. R.M. Murray and S. Sastry, "Steering nonholonomic systems using sinusoids," *IEE Int. Conf. on Decision and Control*, pp. 2097–2101, 1990.
59. R.M. Murray, "Robotic Control and Nonholonomic Motion Planning," PhD Thesis, Memorandum No. UCB/ERL M90/117, University of California, Berkeley, 1990.
60. R.M. Murray, "Nilpotent bases for a class on nonintegrable distributions with applications to trajectory generation for nonholonomic systems," *Math. Control Signal Syst.*, Vol. 7, pp. 58–75, 1994.
61. N.J. Nilsson, "A mobile automaton: an application of artificial intelligence techniques," *1st Int. Joint Conf. on Artificial Intelligence*, pp. 509–520, Washington, 1969.
62. C. O'Dunlaing, "Motion planning with inertial constraints," *Algorithmica*, Vol. 2 (4), 1987.
63. P. Rouchon, M. Fliess, J. Lévine and P. Martin, "Flatness and motion planning: the car with n trailers," *European Control Conference*. pp. 1518–1522, 1993.
64. P. Rouchon, "Necessary condition and genericity of dynamic feedback linearization," in *J. Math. Systems Estimation Control*, Vol. 4 (2), 1994.
65. J. A. Reeds and R. A. Shepp, "Optimal paths for a car that goes both forward and backwards," *Pacific Journal of Mathematics*, 145 (2), pp. 367–393, 1990.
66. J. Reif and H. Wang, "Non-uniform discretization approximations for kinodynamic motion planning and its applications," *Algorithms for Robotic Motion and Manipulation, WAFR'96*, J.P. Laumond and M. Overmars Eds, A.K. Peters, 1997.
67. M. Renaud and J.Y. Fourquet, "Time-optimal motions of robot manipulators including dynamics," *The Robotics Review 2*, O. Khatib, J.J. Craig and T. Lozano-Pérez Eds, MIT Press, 1992.
68. J.J. Risler, "A bound for the degree of nonholonomy in the plane," *Theoretical Computer Science*, Vol. 157, pp. 129–136, 1996.
69. J.T. Schwartz and M. Sharir, "On the 'Piano Movers' problem II: general techniques for computing topological properties of real algebraic manifolds," *Advances in Applied Mathematics*, 4, pp. 298–351, 1983.
70. S. Sekhavat, P. Švestka, J.P. Laumond and M. H. Overmars, "Multi-level path planning for nonholonomic robots using semi-holonomic subsystems," *Algorithms for Robotic Motion and Manipulation, WAFR'96*, J.P. Laumond and M. Overmars Eds, A.K. Peters, 1997.

71. S. Sekhavat, "Planification de mouvements sans collisions pour systèmes non holonomes," PhD Thesis 1240, INPT, LAAS-CNRS, Toulouse, 1996.
72. S. Sekhavat and J-P. Laumond, "Topological Property of Trajectories Computed from Sinusoidal Inputs for Chained Form Systems," *IEEE Int. Conf. on Robotics and Automation*, Mineapollis, 1996.
73. S. Sekhavat and J-P. Laumond, "Topological property for collision-free nonholonomic motion planning: the case of sinusoidal inputs for chained form systems," *IEEE Transaction on Robotics and Automation* (to appear).
74. S. Sekhavat, F. Lamiraux, J-P. Laumond, G. Bauzil and A. Ferrand, "Motion planning and control for Hilare pulling a trailer: experimental issues," in *IEEE Int. Conf. on Robotics and Automation*, Albuquerque, 1997.
75. J.J.E. Slotine and H.S. Yang, "Improving the efficiency of time-optimal path-following algorithms," *IEEE Transactions on Robotics and Automation*, 5 (1), pp.118-124, 1989.
76. P. Souères and J.P. Laumond, "Shortest path synthesis for a car-like robot," *IEEE Trans. on Auromatic Control*, Vol. 41 (5), pp. 672–688, 1996.
77. E.D. Sontag, "Controllability is harder to decide than accessibility," *SIAM J. Control and Optimization*, Vol. 26 (5), pp. 1106–1118, 1988.
78. O.J. Sordalen, "Conversion of a car with n trailers into a chained form," *IEEE Int. Conf. on Robotics and Automation*, pp. 382–387, Atlanta, 1993.
79. S. Sternberg, *Lectures on Differential Geometry*, Chelsea Pub., 1983.
80. R. S. Strichartz, "Sub-Riemannian geometry," *Journal of Differential Geometry*, Vol. 24, pp. 221–263, 1986.
81. R. S. Strichartz, "The Campbell-Baker-Hausdorff-Dynkin formula and solutions of differential equations," *Journal of Functional Analysis*, Vol. 72, pp. 320–345, 1987.
82. H.J. Sussmann and V. Jurdjevic, " Controllability of nonlinear systems," *J. of Differential Equations*, 12, pp. 95–116, 1972.
83. H. Sussmann, "Lie brackets, real analyticity and geometric control," *Differential Geometric Control Theory* (R. Brockett, R. Millman and H. Sussmann, eds.), Vol. 27 of *Progress in Mathematics*, pp. 1–116, Michigan Technological University, Birkhauser, 1982.
84. H. J. Sussmann and W. Liu, "Limits of highly oscillatory controls and the approximation of general paths by admissible trajectories," Tech. Rep. SYSCON-91-02, Rutgers Center for Systems and Control, 1991.
85. P. Svestka and M. Overmars, "Coordinated motion planning for multiple car-like robots using probabilistic roadmaps," *IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, 1995.
86. D.Tilbury, R. Murray and S. Sastry, "Trajectory generation for the n-trailer problem using Goursat normal form," *IEEE Trans. on Automatic Control*, Vol. 40 (5), pp. 802–819, 1995.
87. D. Tilbury, J.P. Laumond, R. Murray, S. Sastry and G. Walsh, "Steering car-like systems with trailers using sinusoids," in *IEEE Conf. on Robotics and Automation*, pp. 1993–1998, Nice, 1992.
88. A. Thompson, "The navigation system of the JPL robot," *5th Int. Joint Conf. on Artificial Intelligence*, pp. 749–757, Cambridge, 1977.

89. P. Tournassoud, " Motion planning for a mobile robot with a kinematic constraint," *Geometry and Robotics*, J.D. Boissonnat and J.P. Laumond Eds, pp. 150–171, Lecture Notes in Computer Science, Vol 391, Springer Verlag, 1989.
90. V.S. Varadarajan, *Lie Groups, Lie Algebra and their Representations*, Springer-Verlag, 1984.
91. M. Vendittelli and J.P. Laumond, "Visible positions for a car-like robot amidst obstacles," *Algorithms for Robotic Motion and Manipulation*, J.P. Laumond and M. Overmars Eds, A.K. Peters, 1997.
92. A.M. Vershik and V.Ya. Gershkovich, "Nonholonomic problems and the theory of distributions," *Acta Applicandae Mathematicae*, Vol. 12, pp. 181–209, 1988.
93. X. Viennot, *Algèbres de Lie libres et monoïdes libres.* Lecture Notes in Mathematics, 691, Springer Verlag, 1978.
94. G.T. Wilfong, " Motion planning for an autonomous vehicle," *IEEE Int. Conf. on Robotics and Automation*, pp. 529–533, 1988.

Geometry of Nonholonomic Systems

A. Bellaïche[1], F. Jean[2] and J.-J. Risler[3]

[1] Paris 7 University
[2] Paris 6 University
[3] Ecole Normale Supérieure and Paris 6 University

Nonholonomic motion planning is best understood with some knowledge of the underlying geometry. In this chapter, we first introduce in Section 1 the basic notions of the geometry associated to control systems without drift. In the following sections, we present a detailed study of an example, the car with n trailers, then some general results on polynomial systems, which can be used to bound the complexity of the decision problem and of the motion planning for these systems.

1 Symmetric control systems: an introduction

1.1 Control systems and motion planning

Regardless of regularity hypotheses, control systems may be introduced in two ways. By ascribing some condition

$$\dot{x} \in V_x$$

where V_x is, for every x, some subset of the tangent space T_xM, or in a parametric way, as

$$\dot{x} = f(x, u)$$

where, for every x, the map $u \mapsto f(x, u)$ has V_x as its image.

In mechanics or robotics, conditions of the first kind occur as linear constraints on the velocities, such as rolling constraints, as well in free movement—the classical object of study in mechanics, as in the case of systems propulsed by motors.

Equations of the second kind may represent the action of "actuators" used to move the state of the system in the configuration space. One can show that if the action of two actuators are represented by $\dot{x} = f_1(x)$ and $\dot{x} = f_2(x)$, we may also consider the action of any convex combination of vector fields f_1 and f_2, and add it to the possible actions without changing in an essential way the accessible set $A(x)$ or $A(x, T)$. For this reason, one may suppose V_x to be convex, or equivalently, $u \mapsto f(x, u)$ to be affine, of the form $(u_1, \ldots, u_m) \mapsto$

$X_0(x) + u_1X_1(x) + \cdots + u_mX_m(x)$, and defined on some convex subset K_x of $\mathbf{R}^m$, for some m. This is responsible for the form

$$\dot{x} = X_0(x) + u_1X_1(x) + \cdots + u_mX_m(x)$$

under which control systems are often encountered in the literature. (It makes no harm to suppose m and K_x to be independent of x, and to suppose that the origin is an interior point of $K = K_x$.) The vector field X_0 is called the drift.

Now, we will use only systems without drift, that is with $X_0 = 0$, for the study of the problem of motion planning for robots. We may content with such systems as long as no dynamics is involved. That is, if the state of the system represents its position, and if we control directly its velocity. As opposed to a system whose state would represent position *and* velocities, and where the control is exerted on accelerations. Consider the simplest possible of such a system: a mobile point on a line, submitted to the control equation

$$\ddot{x} = u.$$

Introducing the velocity $y = \dot{x}$, we see that this system is equivalent to a system governed by the equation

$$\dot{x} = y$$
$$\dot{y} = u$$

which can be written as

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix} + u \begin{pmatrix} 0 \\ 1 \end{pmatrix} = X_0 + uX_1,$$

that is, with a non-zero drift X_0.

For some applications, our study will be valid in the case of slow motion only, and resemble to the thermodynamics of equilibriums, where all transformation are supposed to be infinitely slow.

1.2 Definitions. Basic problems

To sum up, we shall be interested in control systems of the form

$$\dot{x} = \sum_{i=1}^{m} u_iX_i(x), \qquad x \in M, \tag{Σ}$$

where the configuration space M of the system is a C^∞ manifold, $X_1, \ldots, X_m$ are C^∞ vector fields on M, and the control function $u(t) = (u_1(t), \ldots, u_t(t))$ takes values in a fixed compact convex K of $\mathbf{R}^m$, with nonempty interior, and

symmetric with respect to the origin. Such systems are called *symmetric* (or *driftless*). One also says that controls enter *linearly* in (Σ).

For any choice of u as a measurable function defined on some interval $[0,T]$, with value in K, equation (Σ) becomes a differential equation

$$\dot{x} = \sum_{i=1}^{m} u_i(t) X_i(x). \tag{1}$$

Given any point x_0 on M, we can integrate (1), taking

$$x(0) = x_0 \tag{2}$$

as an initial condition. For the sake of simplicity, we shall suppose that this equation has a well-defined solution on $[0,T]$ for all choices of u (this is guaranteed if M is compact or if $M = \mathbf{R}^n$, and vector fields X_i are bounded). Call this solution x_u. One says that x_u is the path with initial point x_0 and controlled by u. We shall mainly be interested in its final value $x_u(T)$. Classically, points in M are called the *states* of the system. One says for example that the system is *steered* from state x_0 to state $x_u(T)$ by means of the control function u.

One also says that $x_u(T)$ is *accessible*, or *reachable*, in time T from x_0. We shall denote by $A(x,T)$ the set of points of M accessible from x in time T (or in time $\leq T$, it is the same thing for symmetric systems), and by $A(x)$ the set of points accessible from x, that is

$$A(x) = \bigcup_{T>0} A(x,T).$$

Basic problems of Control Theory are:

- determine the accessible set $A(x)$;
- given a point y, accessible from x, find control functions steering the system from x to y;
- do the preceding in minimal time;
- more generally, find control function u ensuring any given property of $x_u(t)$, the path controlled by u.

Given x_0, the control function $u(t)$ is considered as the *input* of the system, and $x_u(t)$ as the *output*. In a more general setting, the output is only some function $h(x)$ of the state x, h being called the observation: the state is only partially known. Here we will take as observation $h = \mathrm{Id}$, and call indifferently x the state or the output.

We can now state another basic problem:

– can one find a map $k : M \to K$ such that the differential equation

$$\dot{x} = f(x, k(x)) \tag{3}$$

has a determined behaviour, for example, has a given point x_0 as an attractor?

Since in this problem, the output is reused as an input, such a map k is called a *feedback control law*, or a *closed-loop control*. If (3) has x_0 as an attractor, one says that k is a *stabilizing feedback* at x_0.

1.3 The control distance

Return to the control system (Σ). For $x, y \in M$, define $d(x, y)$ as the infimum of times T such that y is accessible from x in time T, so $d(x, y) = +\infty$ if y is not accessible from x. It is immediate to prove that $d(x, y)$ is a distance [distance function] on M. Of course, this is the case only because we supposed that K is symmetric with respect to the origin in $\mathbf{R}^m$.

Distance d will be called the *control distance*.

We can define d in a different way. First, observe that since K is convex, symmetric, with nonempty interior, we can associate to it a norm $\|\cdot\|_K$ on $\mathbf{R}^m$, such that K is the unit ball $\|u\|_K \leq 1$. Now, for a controlled path $c = x_u : [a, b] \to M$ obtained by means of a control function $u \in L^1([a, b], \mathbf{R}^m)$, we set

$$\text{length}(c) = \int_a^b \|u(t)\|_K \, dt. \tag{4}$$

If c can be obtained in such a way from several different u's, we take the infimum of the corresponding integrals. Then, $d(x, y)$ is the infimum of the lengths of controlled paths joining x to y (and, of course, this is intended in the definition of an infimum, $+\infty$ if no such path exists).

A slightly variant construction may be useful. Transfer the function $\|\cdot\|_K$ to $T_x M$, by setting

$$\|v\|_K = \inf\{\, \|(u_1, \ldots, u_m)\|_K \mid v = u_1 X_1(x) + \cdots + u_m X_m(x) \,\}.$$

We get in this way a function on $T_x M$ which is a norm on $\text{span}(X_1(x), \ldots, X_m(x))$ and takes the value $+\infty$ for vector not in this subspace. We can now define the length of any absolutely continuous path $c : [a, b] \to M$ as

$$\text{length}(c) = \int_a^b \|\dot{c}(t)\|_K \, dt$$

and the distance $d(x, y)$ as the infimum of length of paths joining x and y.

Note that distances corresponding to different K, say K_1 and K_2, are equivalent: there exists some positive constants A and B such that

$$Ad_1(x,y) \leq d_2(x,y) \leq Bd_1(x,y).$$

The most convenient version of the control distance is obtained by taking for K the unit ball of $\mathbf{R}^m$, which gives

$$\|u\| = (u_1^2 + \cdots + u_m^2)^{1/2}.$$

In this case, the distance d is called *the sub-Riemannian distance attached to the system of vector fields* $X_1, \ldots, X_m$. As a justification for this name, observe that, locally, any Riemannian distance can be recovered in such a way by taking $m = n$, and as $X_1(x), \ldots, X_n(x)$ an orthonormal basis, depending on x, of the tangent space T_xM. A more general, more abstract, definition of sub-Riemannian metrics can be given, but we shall not use it in this book.

Now, observe that $d(x,y) < \infty$ if and only if x and y are reachable from one another, that $A(x,T)$ is nothing else that the ball of center x and radius T (for d), and that controlled paths joining x to y in minimal time are simply minimizing geodesics.

Many problems of control theory, or path planning, get in this way a geometric interpretation. For another example, one could think to obtain a feedback law $k(x)$ stabilizing the system at $x = x_0$ by choosing k so as to ensure $f(x,k(x))$ to be the gradient of $d(x,x_0)$. Unfortunately, this does not work, even if we take the good version of the gradient, *i.e.*, the sub-Riemannian one:

$$\operatorname{grad} f = (X_1 f)X_1 + \cdots + (X_m f)X_m.$$

and take $k(x) = (X_1 f, \ldots, X_m f)$ for that purpose. But studying the reasons of this failure is very instructive. Such a geometric interpretation, using the sub-Riemaniann distance, really brings a new insight in theory, and it will in several occasions be very useful to us.

1.4 Accessibility. The theorems of Chow and Sussmann

We shall deduce the classical theorem of Chow (Chow [7], Rashevskii [28]) from a more precise result by Sussmann. Sussmann's theorem will be proved using L^1 controls. However, it can be shown that the results obtained are, to a great extend, independent of the class of control used (see Bellaïche [2]).

Consider a symmetric control system, as described above,

$$\dot{x} = \sum_{i=1}^{m} u_i X_i(x), \qquad x \in M, \quad u \in K. \tag{Σ}$$

Recall the configuration space M is a C^∞ manifold, $X_1, \ldots, X_m$ are C^∞ vector fields on M, and K, the control set, or parameter set, is a fixed compact convex of $\mathbf{R}^m$, with nonempty interior, *symmetric* with respect to the origin.

In all this section, we fix a point $x_0 \in M$, the initial point, and a positive time T. Set

$$\mathcal{H}_T = L^1([0,T], \mathbf{R}^m).$$

We shall call this space the *space of controls*. It may be considered as a normed space by setting

$$\|u\| = \int_0^T \|u(t)\|_K \, dt.$$

Given $u \in \mathcal{H}_T$, we consider the differential equation

$$\begin{cases} \dot{x} = \sum_{i=1}^m u_i(t) X_i(x), \; 0 \leq t \leq T \\ x(0) = x_0 \end{cases} \tag{5}$$

Under suitable hypotheses, the differential equation (5) has a well defined solution $x_u(t)$. We will denote by

$$\mathrm{End}_{x_0,T} : \mathcal{H}_T \to M$$

the mapping wich maps u to $x_u(T)$. We will call $\mathrm{End}_{x_0,T}$, or End for short, the end-point map.

Now, the accessible set $A(x_0)$ (the set of points accessible from x_0 for the system Σ, regardless of time) is exactly the image of $\mathrm{End}_{x_0,T}$. Indeed, every controlled path $c : [0,T'] \to M$, defined by the control $u : [0,T'] \to M$ may be reparametrized by $[0,T]$. Conversely, if $u \in \mathcal{H}$, and $L = \mathrm{length}(x_u)$, the control function

$$v(t) = \frac{u(\phi(t))}{\|u(\phi(t))\|_K}, \qquad 0 \leq t \leq L,$$

where ϕ is defined as a right inverse to the mapping

$$s \mapsto \int_0^s \|u(\tau)\|_K \, d\tau$$

from $[0,T]$ to $[0,L]$ takes its values in K, and defines the same geometric path as u.

Theorem 1.1 (Sussmann [36], Stefan [35]). *The set $A(x_0)$ of points accessible from a given point x_0 in M is an immersed submanifold.*

We shall prove this theorem using arguments from differential calculus in Banach spaces, taking advantage from the fact that the end-point map is a differentiable mapping from $\mathcal{H}$ to M, a finite dimensional manifold.

Recall the rank of a differentiable mapping at a given point is by definition the rank of its differential at that point. The theorem of the constant rank asserts that the image of a differential map with constant rank is an immersed submanifold (for more details about this part of the proof, see Bellaïche [2]).

Definition. Let ρ the maximal rank of the end-point map $\mathrm{End}_{x_0,T} : \mathcal{H}_T \to M$. We say that a control function $u \in \mathcal{H}_T$ is *normal* if the rank of $\mathrm{End}_{x_0,T}$ at u is equal to ρ. We shall say that the path x_u defined by u is a normal path. Otherwise, u is said to be an *abnormal* control, and x_u an abnormal path. A point which can be joined to x_0 by a normal path is said to be *normally accessible from* x_0.

Lemma 1.2. *Every point accessible from x_0 is normally accessible from x_0.*

Proof. Let y be a point accessible from x_0, and let $u \in \mathcal{H}_T$ a control steering x_0 to y. Choose a normal control $z \in \mathcal{H}_T$, steering x_0 to some point z. Such a control exists by definition. We claim that the control function $w \in \mathcal{H}_{3T}$ defined by

$$w(t) = \begin{cases} v(t) & \text{if } 0 \le t \le T \\ v(2T - t) & \text{if } T \le t \le 2T \\ u(t - 2T) & \text{if } 2T \le t \le 3T \end{cases}$$

is normal and steers x_0 to y.

The second part of our assertion is evident: the path x_w steers x_0, first to z, then back to x_0, then to y. Now, the image of $D\,\mathrm{End}_{x_0,3T}$ consists of the infinitesimal variations $\delta x_w(3T)$ obtained from infinitesimal variations δw of w. We can consider special variations of w, namely variations of the first part of w only, leaving the two other parts unchanged. In other words, we consider the control functions

$$w(t) + \delta w(t) = \begin{cases} v(t) + \delta v(t) & \text{if } 0 \le t \le T \\ v(2T - t) & \text{if } T \le t \le 2T \\ u(t - 2T) & \text{if } 2T \le t \le 3T \end{cases}$$

Since v is a normal control, these variations yield infinitesimal variations of $\delta x_w(T) = x_{w+\delta w}(T) - x_w(T)$ which form a subspace of dimension ρ at that point. Now, the corresponding variations of $x_w(3T)$ are obtained from those of $x_w(T)$ by applying the flow of the time-dependent vector field $\sum_{1 \le i \le m} w_i(t) X_i(x)$ between time T and time $3T$. Since this flow is a diffeomorphism of M, these variations of $x_w(3T)$ form a subspace of dimension ρ of the tangent space $T_y M$. The space formed by variations of the $x_w(3T)$ caused by unrestricted variations of the control w has thus dimension $\ge \rho$, an so has dimension ρ. This proves that w is normal.

Of course, the fact that w is in $\mathcal{H}_{3T}$ instead of being in $\mathcal{H}_T$ is harmless. ∎

Proof of Sussmann's theorem. The normal controls form an open subset $N_{x_0,T}$ of $\mathcal{H}_T$. From Theorem 1.2, the accessible set $A(x_0)$ is the image of $N_{x_0,T}$ by a constant rank map. By using the Theorem of constant rank, the proof is done. ∎

Theorem 1.3 (Chow [7], Rashevskii [28]). *If M is connected (for its original topology), and if the vector fields $X_1, \ldots, X_m$ and their iterated brackets $[X_i, X_j]$, $[[X_i, X_j], X_k]$, etc. span the tangent space T_xM at every point of M, then any two points of M are accessible from one another.*

Proof. Since the relation $y \in A(x)$ is clearly an equivalence relation, we can speak of *accessibility components.* Since

$$y \in A(x) \Longleftrightarrow d(x, y) < \infty,$$

the set $A(x)$ is the union of open balls $B(x, R)$ (for d), so it is itself an open set. Whence it results that the accessibility components are also the connected component of M for the topology defined by d.

It is clear that the accessibility components of M are stable under the flow $\exp tX_i$ of vector field X_i $(i = 1, \ldots, m)$. Therefore, the vector fields $X_1, \ldots, X_m$ are, at any point, tangent to the accessibility component through that point (see [2] for details). And so are their brackets $[X_i, X_j]$, their iterated brackets $[[X_i, X_j], X_k]$, etc.

If the condition on the brackets is fulfilled, then

$$T_xA(x) = T_xM$$

at every point x, as the preceding discussion shows. In that case, the accessibility components are open. Since M is connected, there can be only one accessibility component. ∎

Definition. The following condition

(C) The vector fields $X_1, \ldots, X_m$ and their iterated brackets $[X_i, X_j]$, $[[X_i, X_j], X_k]$, etc. span the tangent space T_xM at every point of M,

is called Chow's Condition.

When the Chow's Condition holds, one says that system (Σ) is *controllable.* The reciprocal of Chow's theorem, that is, if (Σ) is controllable, the X_i's and their iterated brackets span the tangent space at every point of M, is true if M and the vector fields are analytic, and false in the C^∞ case (see Sussmann [36]).

Chow's Condition is also known under the name of Lie Algebra Rank Condition (LARC) since it states that the rank at every point x of the Lie algebra generated by the X_i's is full (self-evident definition). In the context of PDE, it is known under the name of Hörmander's Condition: if it is verified, the differential operator $X_1^2 + \cdots + X_m^2$ is hypoelliptic (Hörmander's Theorem).

1.5 The shape of the accessible set in time ε

The purpose of this section is to study the geometric structure of $A(x,\varepsilon)$ for small ε. Let us recall that $A(x,\varepsilon)$ denotes the set of points accessible from x in time ε (or in time $\leq \varepsilon$, it is the same thing) by means of control u_i such that $\sum u_i^2 \leq 1$. In other words, $A(x,\varepsilon)$ is equal to $B(x,\varepsilon)$, the sub-Riemaniann closed ball centered at x with radius ε.

We suppose in the sequel that Chow's condition is satisfied for the control system (Σ). Choosing some chart in a neighbourhood of x_0, we may write (1) as

$$\dot{x} = \sum_{i=1}^{m} u_i(t)\Big(X_i(0) + O(\|x\|)\Big) \tag{6}$$

The differential equation (1) thus appear as a perturbation of the trivial equation

$$\dot{x} = \sum_{i=1}^{m} u_i(t) X_i(0) \tag{7}$$

Classical arguments on perturbation of differential equations show that the solution of (6) is given by

$$x(T) = x(0) + \sum_{i=1}^{m}\Big(\int_0^T u_i(t)\,dt\Big)\,X_i(0) + O(\|u\|^2), \tag{8}$$

where, for u, we use the L^1 norm. Thus, with a linear change of coordinates, the set of points accessible from $x(0) = 0$ in time $T \leq \varepsilon$ satisfies, for small ε

$$A(x,\varepsilon) \subset C[-\varepsilon,\varepsilon]^{n_1} \times [-\varepsilon^2,\varepsilon^2]^{n-n_1},$$

where n_1 is the rank of the family $X_1(0),\ldots,X_m(0)$. As a first step, the set $A(x,\varepsilon)$ is then included in a flat pancake.

The expression (8) implies also that the differential of the end-point mapping at the origin in $\mathcal{H}$ is the linear map

$$u \mapsto \sum_{i=1}^{m}\Big(\int_0^T u_i(t)\,dt\Big)\,X_i(0).$$

Since, typically, we suppose $m < n$, this linear map has rank $n_1 < n$ and the end-point mapping is not a submersion at $0 \in \mathcal{H}$. Following our definition, this means that the constant path at x_0 is an abnormal path. This result has a lot of consequences.

Given a neighbourhood U of x_0, there may not exist a *smooth* mapping $x \mapsto u^x$ of U into $\mathcal{H}$ such that the control u^x steers x_0 to x, or, as well, x to x_0. A stronger result is Brockett's theorem asserting the non-existence of a continuous feedback law, stabilizing system (Σ) at a given point x_0, when $m < n$.

To go further in the description of the set $A(x, \varepsilon)$ we can use the so-called *iterated integrals.* For example, the system

$$\begin{cases} \dot{x}_1 = u_1 \\ \dot{x}_2 = u_2 \\ \dot{x}_3 = x_1 u_2 - x_2 u_1 \\ x_1(0) = x_2(0) = x_3(0) = 0 \end{cases} \tag{9}$$

is solved by

$$\begin{aligned} x_1(T) &= \int_0^T u_1(t)\,dt \\ x_2(T) &= \int_0^T u_2(t)\,dt \\ x_3(T) &= \int_0^T \left(\int_0^{t_1} u_1(t_2)\,dt_2 \right) u_2(t_1)\,dt_1 - \int_0^T \left(\int_0^{t_1} u_2(t_2)\,dt_2 \right) u_1(t_1)\,dt_1 \end{aligned} \tag{10}$$

This scheme works for chained or triangular systems, that is, $\dot{x}_j$ depends only on the controls and $x_1, \ldots, x_{j-1}$. But we shall see that it can be put to work for any system. To begin with, let us rewrite (9) as

$$\dot{x} = u_1 X_1(x) + u_2 X_2(x), \quad x(0) = 0.$$

Then (10) can be read as

$$\begin{aligned} x(T) = x(0) &+ \left(\int_0^T u_1(t)\,dt \right) X_1(0) + \left(\int_0^T u_2(t)\,dt \right) X_2(0) + \\ &\left(\int_0^T \left(\int_0^{t_1} u_1(t_2)\,dt_2 \right) u_2(t_1)\,dt_1 - \right. \\ &\left. \int_0^T \left(\int_0^{t_1} u_2(t_2)\,dt_2 \right) u_1(t_1)\,dt_1 \right) [X_1, X_2](0). \end{aligned}$$

Put this way, the formula for $x(T)$ can readily be generalized to any control system of the form

$$\dot{x} = \sum_{1 \le i \le m} u_i X_i(x).$$

One gets (the proof is not hard, cf. Brockett [5])

$$x(T) = x(0) + \sum_{1\le i\le m} \left(\int_0^T u_1(t)\,dt + O(\|u\|^2) \right) X_i(0) + \sum_{1\le i<j\le m} \left(\int_0^T \left(\int_0^{t_1} u_i(t_2)\,dt_2 \right) u_j(t_1)\,dt_1 - \int_0^T \left(\int_0^{t_1} u_j(t_2)\,dt_2 \right) u_i(t_1)\,dt_1 \right) [X_i, X_j](0) + O(\|u\|^3),$$

or, written in a more civilized manner

$$x(T) = x(0) + \sum_{1\le i\le m} (A_i^T(u) + O(\|u\|^2))X_i(0) + \sum_{1\le i<j\le m} A_{ij}^T(u)[X_i, X_j](0) + O(\|u\|^3)$$

which can, for given T, be considered as a limited expansion of order 2 of the end-point mapping about 0 in $\mathcal{H}$. Observe that $A_i^T(u)$ is a linear function with respect to $u \in \mathcal{H}$, and $A_{ij}^T(u)$ is a quadratic function on $\mathcal{H}$. This expansion generalizes the expansion (8) and the set $A(x,\varepsilon)$ satisfies now

$$A(x,\varepsilon) \subset C[-\varepsilon,\varepsilon]^{n_1} \times [-\varepsilon^2,\varepsilon^2]^{n_2-n_1} \times [-\varepsilon^3,\varepsilon^3]^{n-n_2}. \tag{11}$$

Having shown that $A(x,\varepsilon)$ is contained in some box, one can ask whether it contains some other box of the same kind. Of course, before this question can be taken seriously, one has to replace inclusion (11) by

$$A(x,\varepsilon) \subset C[-\varepsilon,\varepsilon]^{n_1} \times [-\varepsilon^2,\varepsilon^2]^{n_2-n_1} \times [-\varepsilon^3,\varepsilon^3]^{n_3-n_2} \times \cdots$$

where the integers n_1, n_2, n_3, ... are the best possible.

Now, except for the case $n_2 = n$ which can be dealt with directly, the proof of an estimate like

$$C'[-\varepsilon,\varepsilon]^{n_1} \times [-\varepsilon^2,\varepsilon^2]^{n_2-n_1} \times [-\varepsilon^3,\varepsilon^3]^{n_3-n_2} \times \cdots \subset A(x,\varepsilon)$$

requires new techniques and special sets of coordinates. Instead of computing limited expansion up to order r, we will compute an expansion to order 1 only, but by assigning weights to the coordinates. This will be done in §§1.6–1.8.

1.6 Regular and singular points

In the sequel we will fix a manifold M, of dimension n, a system of vector fields $X_1,\ldots,X_m$ on M. We will suppose that $X_1,\ldots,X_m$ verify the condition

of Chow. We will denote by d the distance defined on M by means of vector fields $X_1, \ldots, X_m$.

Let $\mathcal{L}^1 = \mathcal{L}^1(X_1, \ldots, X_m)$ be the set of linear combinations, with real coefficients, of the vector fields $X_1, \ldots, X_m$. We define recursively $\mathcal{L}^s = \mathcal{L}^s(X_1, \ldots, X_m)$ by setting

$$\mathcal{L}^s = \mathcal{L}^{s-1} + \sum_{i+j=s} [\mathcal{L}^i, \mathcal{L}^j]$$

for $s = 2, 3, \ldots$, as well as $\mathcal{L}^0 = 0$. The union $\mathcal{L} = \mathcal{L}(X_1, \ldots, X_m)$ of all $\mathcal{L}^s$ is a Lie subalgebra of the Lie algebra of vector fields on M which is called the *control Lie algebra* associated to (Σ).

Now, for p in M, let $L^s(p)$ be the subspace of T_pM which consists of the values $X(p)$ taken, at the point p, by the vector fields X belonging to $\mathcal{L}^s$. Chow's condition states that for each point $p \in M$, there is a smallest integer $r = r(p)$ such that $L^{r(p)}(p) = T_pM$. This integer is called the *degree of nonholonomy* at p. It is worth noticing that $r(q) \leq r(p)$ for q near p. For each point $p \in M$, there is in fact an increasing sequence of vector subspaces, or flag:

$$\{0\} = L^0(p) \subset L^1(p) \subset \cdots \subset L^s(p) \subset \cdots \subset L^{r(p)}(p) = T_pM.$$

We shall denote this flag by $\mathcal{F}(p)$.

Points of the control system split into two categories: regular states, around which the behaviour of the system does not change in a qualitative way, and singular states, where some qualitative changes occur.

Definition. We say that p is a *regular point* if the integers $\dim L^s(q)$ $(s = 1, 2, \ldots)$ remain constant for q in some neighbourhood of p. Otherwise we say that p is a *singular point.*

Let us give an example. Take $M = \mathbf{R}^2$, and

$$X_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, X_2 = \begin{pmatrix} 0 \\ x^k \end{pmatrix}$$

(k is some integer). Then for $c = (x, y)$ we have $\dim L^1(c) = 1$ if $x = 0$, $\dim L^1(c) = 2$ if $x \neq 0$, so all points on the line $x = 0$ are singular and the others are regular. For other examples, arising in the context of mobile robot with trailers, see Section 2.

It is worth to notice that, when M and vector fields $X_1, \ldots, X_m$ are analytic, regular points form an open dense set in M. Moreover, the sequence $\dim L^s(p)$, $s = 0, 1, 2, \ldots$, is the same for all regular points in a same connected component of M and is streactly increasing for $0 \leq s \leq r(p)$. Thus the degree

of nonholonomy at a regular point is bounded by $n-m+1$ (if we suppose that no one of the X_i's is at each point a linear combination of the other vector fields). It may be easily computed when the definition of the X_i's allows symbolic computation, as for an analytic function, being non-zero at the formal level is equivalent to being non-zero at almost every point.

Computing, or even bounding the degree of nonholonomy at singular points is much harder, and motivated, for some part, sections 2 and 3 (see also [9,11,19,24]).

1.7 Distance estimates and privileged coordinates

Now, fix a point p in M, regular or singular. We set $n_s = \dim L^s(p)$ $(s = 0, 1, \ldots, r)$.

Consider a system of coordinates centered at p, such that the differentials $dy_1, \ldots, dy_n$ form a basis of T_p^*M adapted to $\mathcal{F}(p)$ (we will see below how to build such coordinates). If $r = 1$ or 2, then it is easy to prove the following local estimate for the sub-Riemannian distance. For y closed enough to 0, we have

$$d(0, (y_1, \ldots, y_n)) \asymp |y_1| + \cdots + |y_{n_1}| + |y_{n_1+1}|^{1/2} + \cdots + |y_n|^{1/2} \tag{12}$$

where $n_1 = \dim L^1(p)$ (the notation $f(y) \asymp g(y)$ means that there exists constants c, $C > 0$ such that $cg(y) \leq f(y) \leq Cg(y)$). Coordinates $y_1, \ldots, y_{n_1}$ are said to be of weight 1, and coordinates $y_{n_1+1}, \ldots, y_n$ are said to be of weight 2.

In the general case, we define the weight w_j as the smallest integer s such that dy_j is non identically zero on $L^s(p)$. (So that $w_j = s$ if $n_{s-1} < j \leq n_s$.) Then the proper generalization of (12) would be

$$d(0, (y_1, \ldots, y_n)) \asymp |y_1|^{1/w_1} + \cdots + |y_n|^{1/w_n}. \tag{13}$$

It turns out that this estimate is generically *false* as soon as $r \geq 3$. A simple counter-example is given by the system

$$X_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad X_2 = \begin{pmatrix} 0 \\ 1 \\ x^2 + y \end{pmatrix} \tag{14}$$

on $\mathbf{R}^3$. We have

$$L^1(0) = L^2(0) = \mathbf{R}^2 \times \{0\}, \quad L^3(0) = \mathbf{R}^3,$$

so that $y_1 = x$, $y_2 = y$, $y_3 = z$ are adapted coordinates and have weight 1, 1 and 3. In this case, the estimates (13) cannot be true. Indeed, this would imply

$$|z| \leq \text{const.}\big(d(0,(x,y,z))\big)^3,$$

whence

$$\Big|z\big(\exp(tX_2)p\big)\Big| \leq \text{const.}\, t^3,$$

but this is impossible since

$$\frac{d^2}{dt^2}\, z\Big(\exp(tX_2)(p)\Big)\Big|_{t=0} = (X_2^2 z)(p) = 1.$$

However a slight nonlinear change of coordinates allows for (13) to hold. It is sufficient to replace y_1, y_2, y_3 by $z_1 = x$, $z_2 = y$, $z_3 = z - y^2/2$.

In the above example, the point under consideration is singular, but one can give similar examples with regular p in dimension ≥ 4. To formulate conditions on coordinate systems under which estimates like (13) may hold, we introduce some definitions.

Call $X_1 f, \ldots, X_m f$ the *nonholonomic partial derivatives of order* 1 *of* f relative to the considered system (compare to $\partial_{x_1} f, \ldots, \partial_{x_n} f$). Call further $X_i X_j f$, $X_i X_j X_k f$, ... the nonholonomic derivatives of order 2, 3, ... of f.

Proposition 1.4. *For a smooth function f defined near p, the following conditions are equivalent:*

(i) *One has $f(q) = O\big(d(p,q)^s\big)$ for q near p.*
(ii) *The nonholonomic derivatives of order $\leq s-1$ of f vanish at p*

This is proven by the same kind of computations as in the study of example (14).

Definition. If Condition (i), or (ii), holds, we say that f is of order $\geq s$ at p.

Definition. We call local coordinates $z_1, \ldots, z_n$ centered at p a system of *privileged coordinates* if the order of z_j at p is equal to w_j $(j = 1, \ldots, n)$.

If $z_1, \ldots, z_n$ are privileged coordinates, then $dz_1, \ldots, dz_n$ form a basis of $T_p^* M$ adapted to $\mathcal{F}(p)$. The converse is not true. Indeed, if $dz_1, \ldots, dz_n$ form an adapted basis, one can show that the order of z_j is $\leq w_j$, but it may be $< w_j$: for the system (14), the order of coordinate $y_3 = z$ at 0 is 2, while $w_3 = 3$.

To prove the existence, in an effective way, of privileged coordinates, we first choose vector fields $Y_1, \ldots, Y_n$ whose values at p form a basis of $T_p M$ in the following way.

First, choose among $X_1, \ldots, X_m$ a number n_1 of vector fields such that their values form a basis of $L^1(p)$. Call them $Y_1, \ldots, Y_{n_1}$. Then for each s $(s = 2, \ldots, r)$ choose vector fields of the form

$$Y_{i_1 i_2 \ldots i_{s-1} i_s} = [X_{i_1}, [X_{i_2}, \ldots [X_{i_{s-1}}, X_{i_s}] \ldots]] \tag{15}$$

which form a basis of $L^s(p)$ mod $L^{s-1}(p)$, and call them $Y_{n_{s-1}+1}, \ldots, Y_{n_s}$.

Choose now any system of coordinates $y_1, \ldots, y_n$ centered at p such that the differentials $dy_1, \ldots, dy_n$ form a basis dual to $Y_1(p), \ldots, Y_n(p)$. (Starting from any system of coordinates $x_1, \ldots, x_n$ centered at p, one can obtain such a system $y_1, \ldots, y_n$ by a linear change of coordinates.)

Theorem 1.5. *The functions $z_1, \ldots, z_n$ recursively defined by*

$$z_q = y_q - \sum_{\{\alpha \mid w(\alpha) < w_q\}} \frac{1}{\alpha_1! \ldots \alpha_{q-1}!} (Y_1^{\alpha_1} \ldots Y_{q-1}^{\alpha_{q-1}} y_q)(p)\, z_1^{\alpha_1} \ldots z_{q-1}^{\alpha_{q-1}} \tag{16}$$

form a system of privileged coordinates near p. (We have set $w(\alpha) = w_1\alpha_1 + \cdots + w_n\alpha_n$.)

The proof is based on the following lemma.

Lemma 1.6. *For a function f to be of order $> s$ at p, it is necessary and sufficient that*

$$(Y_1^{\alpha_1} \ldots Y_n^{\alpha_n} f)(p) = 0$$

for all $\alpha = (\alpha_1, \ldots, \alpha_n)$ such that $w_1\alpha_1 + \cdots + w_n\alpha_n \leq s$.

This is is an immediate consequence of the following, proved by J.-J. Risler [4]: any product $X_{i_1} X_{i_2} \ldots X_{i_s}$, where $i_1, \ldots, i_s$ are integers, can be rearranged as a sum of ordered monomials

$$\sum c_{\alpha_1 \ldots \alpha_n}(x) Y_1^{\alpha_1} \ldots Y_n^{\alpha_n}$$

with $w_1\alpha_1 + \cdots + w_n\alpha_n \leq s$, and where the $c_{\alpha_1 \ldots \alpha_n}$'s are smooth functions. This result reminds of the Poincaré-Birkhoff-Witt theorem.

Observe that the coordinates $z_1, \ldots, z_n$ supplied by the construction of Theorem 1.5 are given from original coordinates by expressions of the form

$$\begin{aligned} z_1 &= y_1 \\ z_2 &= y_2 + \mathrm{pol}(y_1) \\ &\cdots \\ z_n &= y_n + \mathrm{pol}(y_1, \ldots, y_{n-1}) \end{aligned}$$

where pol denotes a polynomial, without constant or linear term, and that the reciprocal change of coordinates has exactly the same form.

Other ways of getting privileged coordinates are to use the mappings

$$(z_1,\dots,z_n)\mapsto \exp(z_1Y_1+\cdots+z_nY_n)\,p \quad \text{(see [14])},$$
$$(z_1,\dots,z_n)\mapsto \exp(z_nY_n)\cdots\exp(z_1Y_1)\,p \quad \text{(see [18])}.$$

Following the usage in Lie group theory, such coordinates are called canonical coordinates of the first (resp. second) kind.

1.8 Ball-Box Theorem

Using privileged coordinates, the control system (Σ) may be rewritten near p as

$$\dot z_j=\sum_{i=1}^m u_i\big[f_{ij}(z_1,\dots,z_{j-1})+O\big(\|z\|^{w_j}\big)\big]\qquad (j=1,\dots,n),$$

where the functions f_{ij} are weighted homogeneous polynomials of degree w_j-1. By dropping the $O\big(\|z\|^{w_j}\big)$, we get a control system $(\widehat\Sigma)$

$$\dot z_j=\sum_{i=1}^m u_i\big[f_{ij}(z_1,\dots,z_{j-1})\big]\qquad (j=1,\dots,n),$$

or, in short,

$$\dot z=\sum_{i=1}^m u_i\widehat X_i(z),$$

by setting $\widehat X_i=\sum_{j=1}^n f_{ij}(z_1,\dots,z_n)\partial_{z_j}$. This system is nilpotent and the vector fields $\widehat X_i$ are homogeneous of degree -1 under the non-isotropic dilations $(z_1,\dots,z_n)\mapsto(\lambda^{w_1}z_1,\dots,\lambda^{w_n}z_n)$. The system $(\widehat\Sigma)$ is called the *nilpotent homogeneous approximation* of the system (Σ). For the sub-Riemaniann distance $\hat d$ associated to the nilpotent approximation, the estimate (17) below can be shown by homogeneity arguments. The following theorem is then proved by comparing the distances d and $\hat d$ (for a detailed proof, see Bellaïche [2]).

Theorem 1.7. *The estimate*

$$d\,(0,(z_1,\dots,z_n))\asymp |z_1|^{1/w_1}+\cdots+|z_n|^{1/w_n} \tag{17}$$

holds near p if and only if $z_1,\dots,z_n$ form a system of privileged coordinates at p.

The estimate (17) of the sub-Riemannian distance allows to describe the shape of the accessible set in time ε. $A(x,\varepsilon)$ can indeed be viewed as the sub-Riemannian ball of radius ε and Theorem 1.7 implies

$$A(x,\varepsilon) \asymp [-\varepsilon^{w_1},\varepsilon^{w_1}] \times \cdots \times [-\varepsilon^{w_n},\varepsilon^{w_n}].$$

Then $A(x,\varepsilon)$ looks like a box, the sides of the box being of length proportionnal to $\varepsilon^{w_1},\ldots,\varepsilon^{w_n}$. By the fact, Theorem 1.7 is called the Ball-Box Theorem (see Gromov [16]).

1.9 Application to complexity of nonholonomic motion planning

The Ball-Box Theorem can be used to address some issues in complexity of motion planning. The problem of nonholonomic motion planning with obstacle avoidance has been presented in Chapter [Laumond-Sekhavat]. It can be formulated as follows. Let us consider a nonholonomic system of control in the form (Σ). We assume that Chow's Condition is satisfied. The constraints due to the obstacles can be seen as closed subsets F of the configuration space M. The open set $M - F$ is called the *free space*. Let a, $b \in M - F$. The motion planning problem is to find a trajectory of the system linking a and b contained in the free space.

From Chow's Theorem (§1.4), deciding the existence of a trajectory linking a and b is the same thing as deciding if a and b are in the same connected component of $M - F$. Since $M - F$ is an open set, the connexity is equivalent to the arc connexity. Then the problem is to decide the existence of a path in $M - F$ linking a and b. In particular this implies that the decision part of the motion planning problem is the same for nonholonomic controllable systems as for holonomic ones.

For the complete problem, some algorithms are presented in Chapter [Laumond–Sekhavat]. In particular we see that there is a general method (called "Approximation of a collision-free holonomic path"). It consists in dividing the problem in two parts:

- find a path in the free space linking the configurations a and b (this path is called also the collision-free holonomic path);
- approximate this path by a trajectory of the system close enough to be contained in the free space.

The existence of a trajectory approximating a given path can be shown as follows. Choose an open neighbourhood U of the holonomic path small enough to be contained in $M - F$. We can assume that U is connected and then, from Chow's Theorem, there is a trajectory lying in U and linking a and b.

What is the complexity of this method?

The complexity of the first part (*i.e.*, the motion planning problem for holonomic systems) is very well modeled and understood. It depends on the geometric complexity of the environment, that is on the complexity of the geometric primitives modeling the obstacles and the robot in the real world (see [6,30]).

The complexity of the second part requires more developments. It can be seen actually as the "complexity" of the output trajectory. We have then to define this complexity for a trajectory approximating a given path.

Let γ be a collision-free path (provided by solving the first part of the problem). For a given ρ, we denote by $\mathrm{Tube}(\gamma, \rho)$ the reunion of the balls of radius ρ centered at q, for any point q of γ. Let ε be the biggest radius ρ such that $\mathrm{Tube}(\gamma, \rho)$ is contained in the free space. We call ε the *size of the free space around the path* γ. The output trajectories will be the trajectories following γ to within ε, that is the trajectories contained in $\mathrm{Tube}(\gamma, \varepsilon)$.

Let us assume that we have already defined a complexity $\sigma(c)$ of a trajectory c. We denote by $\sigma(\gamma, \varepsilon)$ the infimum of $\sigma(c)$ for c trajectory of the system linking a and b and contained in $\mathrm{Tube}(\gamma, \varepsilon)$. $\sigma(\gamma, \varepsilon)$ gives a complexity of an output trajectory. Thus we can choose it as a definition of the complexity of the second part of our method.

It remains to define the complexity of a trajectory. We will present here some possibilities.

Let us consider first bang-bang trajectories, that is trajectories obtained with controls in the form $(u_1, \ldots, u_m) = (0, \ldots, \pm 1, \ldots, 0)$. For such a trajectory the complexity $\sigma(c)$ can be defined as the number of switches in the controls associated to c.

We will now extend this definition to any kind of trajectory. Following [3], a complexity can be derived from the topological complexity of a real-valued function (*i.e.*, the number of changes in the sign of variation of the function). The complexity $\sigma(c)$ appears then as the total number of sign changes for all the controls associated to the trajectory c. Notice that, for a bang-bang trajectory, this definition coincides with the previous one. We will call *topological complexity* the complexity $\sigma_t(\gamma, \varepsilon)$ obtained with this definition.

Let us recall that the complexity of an algorithm is the number of elementary steps needed to get the result. For the topological complexity, we have chosen as elementary step the construction of a piece of trajectory without change of sign in the controls (that is without manoeuvring, if we think to a car-like robot).

Another way to define the complexity is to use the length introduced in §1.3 (see Formula (4)). For a trajectory c contained in $\mathrm{Tube}(\gamma, \varepsilon)$, we set

$$\sigma_\varepsilon(c) = \frac{\mathrm{length}(c)}{\varepsilon}$$

and we call *metric complexity* the complexity $\sigma_m(\gamma, \varepsilon)$ obtained with $\sigma_\varepsilon(c)$. Let us justify this definition on an example. Consider a path γ such that, for any $q \in \gamma$ and any $i \in \{1, \ldots, m\}$, the angle between $T_q\gamma$ and $X_i(q)$ is greater than a given $\theta \neq 0$. Then, for a bang-bang trajectory without switches contained in $\mathrm{Tube}(\gamma, \varepsilon)$, the length cannot exceed $\varepsilon / \sin\theta$. Thus, the number of switches in a bang-bang trajectory ($\subset \mathrm{Tube}(\gamma, \varepsilon)$) is not greater than the length of the trajectory divided by ε (up to a constant). This links $\sigma_\varepsilon(c)$ and $\sigma_m(\gamma, \varepsilon)$ to the topological complexity.

Let us give an estimation of these complexities for the system of the car-like robot (see Chapter [Laumond–Sekhavat]). The configurations are parametrized by $q = (x, y, \theta)^T \in \mathbf{R}^2 \times \mathcal{S}^1$ and the system is given by:

$$\dot{q} = u_1 X_1 + u_2 X_2, \quad \text{with} \quad X_1 = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix}, \; X_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

It is well-known that, for all $q \in \mathbf{R}^2 \times \mathcal{S}^1$, the space $L^2(q)$ has rank 3 (see Section 2).

Let us consider a non-feasible path $\gamma \subset \mathbf{R}^2 \times \mathcal{S}^1$. When γ is C^1 and its tangent vector is never in $L^1(q)$, one can link the complexity $\sigma_m(\gamma, \varepsilon)$ to the number of ε-balls needed to cover γ. By the Ball-Box Theorem (§1.8), this number is greater than $K\varepsilon^{-2}$, where the constant depends on γ.

More precise results have been proven by F. Jean (see also [22] for weaker estimates). Let $T(q)$ ($\|T\| = 1$) be the tangent vector to γ. Assume that $T(q)$ belongs to $L^2(q) - L^1(q)$ almost everywhere and that γ is parametrized by its arclength s. Then we have, for small $\varepsilon \neq 0$:

$$\sigma_t(\gamma, \varepsilon) \text{ and } \sigma_m(\gamma, \varepsilon) \asymp \varepsilon^{-2} \int_0^L \det(X_1, X_2, T)(\gamma(s))\, ds$$

(let us recall that the notation $\sigma(\gamma, \varepsilon) \asymp f(\gamma, \varepsilon)$ means that there exist c, $C > 0$ independant on γ and ε such that $cf(\gamma, \varepsilon) \leq \sigma(\gamma, \varepsilon) \leq Cf(\gamma, \varepsilon)$).

2 The car with n trailers

2.1 Introduction

This section is devoted to the study of an example of nonholonomic control system: the car with n trailers. This system is nonholonomic since it is subject

to non integrable constraints, the rolling without skiding of the wheels. The states of the system are given by two planar coordinates and $n+1$ angles: the configuration space is then $\mathbf{R}^2 \times (\mathcal{S}^1)^{n+1}$, a $(n+3)$-dimensional manifold. There are only two inputs, namely one tangential velocity and one angular velocity which represent the action on the steering wheel and on the accelerator of the car.

Historically the problem of the car is important, since it is the first nonholonomic system studied in robotics. It has been intensively treated in many papers throughout the litterature, in particular from the point of view of finding stabilizing control laws: see e.g. Murray and Sastry ([25]), Fliess *et al.* ([8]), Laumond and Risler ([23]).

We are interested here in the properties of the control system (see below §2.2). The first question is indeed the controllability. We will prove in §2.4 that the system is controllable at each point of the configuration space. The second point is the study of the degree of nonholonomy. We will give in §2.6 an upper bound which is exponential in terms of the number of trailers. This bound is the sharpest one since it is a maximum. We give also the value of the degree of nonholonomy at the regular points (§2.5). The last problem is the singular locus. We have to find the set of all the singular points (it is done in §2.5) and also to determinate its structure. We will see in §2.7 that one has a natural stratification of the singular locus related to the degree of nonholonomy.

2.2 Equations and notations

Different representations have been used for the car with n trailers. The problem is to choose the variables in such a way that simple induction relation may appear. The kinematic model introduced by Fliess [8] and Sørdalen [33] satisfies this condition. A car in this context will be represented by two driving wheels connected by an axle. The kinematic model of a car with two degrees of freedom pulling n trailers can be given by:

$$\begin{aligned}
\dot{x} &= \cos\theta_0 v_0, \\
\dot{y} &= \sin\theta_0 v_0, \\
\dot{\theta}_0 &= \sin(\theta_1 - \theta_0)\tfrac{v_1}{r_1}, \\
&\vdots \\
\dot{\theta}_i &= \sin(\theta_{i+1} - \theta_i)\tfrac{v_{i+1}}{r_{i+1}}, \\
&\vdots \\
\dot{\theta}_{n-1} &= \sin(\theta_n - \theta_{n-1})\tfrac{v_n}{r_n}, \\
\dot{\theta}_n &= \omega,
\end{aligned} \tag{18}$$

where the two inputs of the system are the angular velocity ω of the car and its tangential velocity $v = v_n$. The state of the system is parametrized by $q = (x, y, \theta_0, \ldots, \theta_n)^T$ where:

- (x, y) are the coordinates of the center of the axle between the two wheels of the *last* trailer,
- θ_n is the orientation angle of the pulling car with respect to the x-axis,
- θ_i, for $0 \leq i \leq n-1$, is the orientation angle of the trailer $(n-i)$ with respect to the x-axis.

Finally r_i is the distance from the wheels of trailer $n-i+1$ to the wheels of trailer $n-i$, for $1 \leq i \leq n-1$, and r_n is the distance from the wheels of trailer 1 to the wheels of the car.

The point of this representation is that the system is viewed from the last trailer to the car: the numbering of the angles is made in this sense and the position coordinates are those of the last trailer. The converse notations would be more natural but unfortunately it would lead to complicated computations.

The tangential velocity v_i of the trailer $n-i$ is given by:

$$v_i = \prod_{j=i+1}^{n} \cos(\theta_j - \theta_{j-1})\, v,$$

or $v_i = f_i v$ where

$$f_i = \prod_{j=i+1}^{n} \cos(\theta_j - \theta_{j-1}).$$

The motion of the system is then characterized by the equation:

$$\dot{q} = \omega X_1(q) + v X_2(q)$$

with the control system $\{X_1, X_2\}$ given by:

$$X_1 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \quad X_2 = \begin{pmatrix} \cos\theta_0\ f_0 \\ \sin\theta_0\ f_0 \\ \vdots \\ \frac{1}{r_n}\sin(\theta_n - \theta_{n-1}) \\ 0 \end{pmatrix}$$

2.3 Examples: the car with 1 and 2 trailers

Let us study first the example of the car with one trailer. The state is $q = (x, y, \theta_0, \theta_1)^T$ and the vector fields are:

$$X_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad X_2 = \begin{pmatrix} \cos\theta_0 \cos(\theta_1 - \theta_0) \\ \sin\theta_0 \cos(\theta_1 - \theta_0) \\ \frac{1}{r_1} \sin(\theta_1 - \theta_0) \\ 0 \end{pmatrix}$$

We want to solve the three problems above (controllability, degree of nonholonomy, singular set). For that, we have to study the Lie Algebra generated by the control system (see §1.6). Let us compute the first brackets of X_1 and X_2:

$$[X_1, X_2] = \begin{pmatrix} -\cos\theta_0 \sin(\theta_1 - \theta_0) \\ -\sin\theta_0 \sin(\theta_1 - \theta_0) \\ \frac{1}{r_1} \cos(\theta_1 - \theta_0) \\ 0 \end{pmatrix}, [X_2, [X_1, X_2]] = \frac{1}{r_1} \begin{pmatrix} \sin\theta_0 \\ -\cos\theta_0 \\ -\frac{1}{r_1} \\ 0 \end{pmatrix}.$$

It is straightforward that, for any q, the vectors $X_1(q)$, $X_2(q)$, $[X_1, X_2](q)$ and $[X_2, [X_1, X_2]](q)$ are independant. This implies that, for each q:

$$\begin{aligned} \dim L_1(X_1, X_2)(q) &= 2, \\ \dim L_2(X_1, X_2)(q) &= 3, \\ \dim L_3(X_1, X_2)(q) &= 4, \end{aligned}$$

where $L_k(X_1, X_2)(q)$ is the linear subspace generated by the values at q taken by the brackets of X_1 and X_2 of length $\leq k$.

These dimensions allow us to resolve our three problems. First, the conditions of the Chow theorem are satisfied at each point (since the configuration space is 4-dimensional), so the car with one trailer is controllable. On the other hand, the dimensions of the $L_k(X_1, X_2)(q)$ doesn't depend on q, so all the points are regular and the degree of nonholonomy is always equal to 3.

Let us consider now the car with 2 trailers. If we compute the first brackets, we obtain the following results:

- if $\theta_2 - \theta_1 \neq \pm\frac{\pi}{2}$, then the first independant brackets are $X_1(q)$, $X_2(q)$, $[X_1, X_2](q)$, $[X_2, [X_1, X_2]](q)$ and $[X_2, [X_2, [X_1, X_2]]](q)$;
- if $\theta_2 - \theta_1 = \pm\frac{\pi}{2}$, then the first independant brackets are $X_1(q)$, $X_2(q)$, $[X_1, X_2](q)$, $[X_2, [X_1, X_2]](q)$ and $[X_1[X_2, [X_2, [X_1, X_2]]]](q)$.

Thus the car with 2 trailers is also controllable since, in both cases, the subspace $L_5(X_1, X_2)(q)$ is 5-dimensional. However we have now a singular set, the points q such that $\theta_2 - \theta_1 = \pm\frac{\pi}{2}$. At these points, the degree of nonholonomy equals 5 and at the regular points it equals 4.

2.4 Controllability

The controllability of the car with n trailers has first been proved by Laumond ([21]) in 1991. He used the kinematic model (18) but a slightly different parametrization where the equation were given in terms of $\varphi_i = \theta_i - \theta_{i-1}$ and (x', y') ((x', y') is the position of the pulling car). The proof of the controllability given here is an adaptation of the proof of Laumond for our parametrization. This adaptation has been presented by Sørdalen ([33]).

Theorem 2.1. *The kinematic model of a car with n trailers is controllable.*

Proof. Let us recall some notations introduced in §1.6.

Let $\mathcal{L}_1(X_1, X_2)$ be the set of linear combinations with real coefficients of X_1 and X_2. We define recursively the distribution $\mathcal{L}_k = \mathcal{L}_k(X_1, X_2)$ by:

$$\mathcal{L}_k = \mathcal{L}_{k-1} + \sum_{i+j=k} [\mathcal{L}_i, \mathcal{L}_j] \tag{19}$$

where $[\mathcal{L}_i, \mathcal{L}_j]$ denotes the set of all brackets $[V, W]$ for $V \in \mathcal{L}_i$ and $W \in \mathcal{L}_j$. The union $\mathcal{L}(X_1, X_2)$ of all $\mathcal{L}_k(X_1, X_2)$ is the Control Lie Algebra of the system $\{X_1, X_2\}$.

Let us now denote $\mathcal{L}'_1(X_1, X_2)$ the set of linear combinations of X_1 and X_2 which coefficients are *smooth functions*. By the induction (19) we construct from $\mathcal{L}'_1(X_1, X_2)$ the sets $\mathcal{L}'_k(X_1, X_2)$ and $\mathcal{L}'(X_1, X_2)$.

For a given state q, we denote by $L_k(X_1, X_2)(q)$, resp. $L'_k(X_1, X_2)(q)$, the subspace of $T_q(\mathbf{R}^2 \times (\mathcal{S}^1)^{n+1})$ wich consists of the values at q taken by the vector fields belonging to $\mathcal{L}_k(X_1, X_2)$, resp. $\mathcal{L}'_k(X_1, X_2)$.

Obviously, the sets $\mathcal{L}_k(X_1, X_2)$ and $\mathcal{L}'_k(X_1, X_2)$ are different. However, for each $k \geq 1$ and each q, the linear subspaces $L_k(X_1, X_2)(q)$ and $L'_k(X_1, X_2)(q)$ are equal. We are going to prove this equality for $k = 2$ (the proof for any k can be easily deduced from this case).

By definition $L_2(X_1, X_2)(q_0)$ is the linear subspace generated by $X_1(q_0)$, $X_2(q_0)$ and $[X_1, X_2](q_0)$. $L'_2(X_1, X_2)(q_0)$ is generated by $X_1(q_0)$, $X_2(q_0)$ and all the $[f(q)X_1, g(q)X_2](q_0)$ with f and g smooth functions. Then $L_2(X_1, X_2)(q_0) \subset L'_2(X_1, X_2)(q_0)$.

From the other hand a bracket $[fX_1, gX_2](q_0)$ is equal to:

$$fg[X_1, X_2](q_0) - g(X_2.f)X_1(q_0) + f(X_1.g)X_2(q_0).$$

Thus $[fX_1, gX_2](q_0)$ is a linear combination with real coefficients of $X_1(q_0)$, $X_2(q_0)$ and $[X_1, X_2](q_0)$. Then $L'_2(X_1, X_2)(q_0) = L_2(X_1, X_2)(q_0)$, which prove our statement for $k = 2$.

To establish the controllability, we want to apply Chow's theorem (see §1.4): we have then to show that the dimension of $L(X_1, X_2)(q)$ is $n + 3$. For that,

we are going to prove that $L'(X_1, X_2)(q)$ is $n+3$-dimensionnal and use the relation $L'(X_1, X_2)(q) = L(X_1, X_2)(q)$.

Let us introduce the following vector fields, for $i \in \{0, \ldots, n-1\}$, which belong to $\mathcal{L}'(X_1, X_2)$:

$$\begin{array}{ll} W_0 = X_1 & W_{i+1} = r_{i+1}(\sin\varphi_i V_i + \cos\varphi_i Z_i) \\ V_0 = X_2 & V_{i+1} = \cos\varphi_i V_i - \sin\varphi_i Z_i) \\ Z_0 = [X_1, X_2] & Z_{i+1} = [W_{i+1}, V_{i+1}]. \end{array}$$

The form of these vector fields can be computed by induction. We give only the expression of the interesting ones:

$$\begin{cases} W_i = (\underbrace{0, \ldots, 0}_{n-i+2}, 1, \underbrace{0, \ldots, 0}_{i})^T, \ i = 0, \ldots, n \\ V_n = (\cos\varphi_0, \frac{1}{r_1}\sin\varphi_0, 0, \ldots, 0)^T, \\ Z_n = (-\sin\varphi_0, \frac{1}{r_1}\cos\varphi_0, 0, \ldots, 0)^T. \end{cases} \tag{20}$$

We have $n+3$ vector fields which values at each point of the configuration space are independant since their determinant equals $1/r_1$. Therefore $L'(X_1, X_2)(q)$, and then $L(X_1, X_2)(q)$, are equal to $T_q(\mathbf{R}^2 \times (\mathcal{S}^1)^{n+1})$. We can then apply Chow's theorem and get the result. ∎

Remark. A stronger concept than controllability is given by the following definition: the system $\{X_1, X_2\}$ is called well-controllable if there exists a basis of $n+3$ vector fields in $L(X_1, X_2)(q)$ such that the determinant of the basis is constant for each point q of the configuration space.
The $n+3$ vector fields that we have constructed in the proof satisfy this condition. So the car with n trailers is well-controllable.

2.5 Regular points

Let us denote $\beta^n(q)$ the degree of nonholonomy of the car with n trailers. It can be defined as:

$$\beta^n(q) = \min\{k \mid \dim L_k(X_1, X_2)(q) = n+3\}.$$

We have already computed (§2.3) the values of this degree for $n = 1$ and 2:

$$\beta^1(q) = 3, \ \beta^2(q) = 4 \text{ or } 5.$$

It appears, for $n = 2$, that the configurations where the car and the first trailer are perpendicular have particular properties. This fact can be generalized as follows ([19]):

Theorem 2.2. *The singular locus of the system is the set of the points for which there exists* $k \in [2,n]$ *such that* $\theta_k - \theta_{k-1} = \pm\frac{\pi}{2}$.

The regular points are then the configuration where no two consecutives trailers (except maybe the last two) are perpendicular. It results from §1.6 that the degree of nonholonomy at regular points is $\leq n+2$. In fact this degree is exactly $n+2$. It can be shown for instance by converting the system into the so-called chained form as in Sørdalen ([33]). This gives us a first result on the degree of nonholonomy:

Theorem 2.3. *At a regular point,* i.e., *a point such that* $\theta_k - \theta_{k-1} \neq \pm\frac{\pi}{2}$ $\forall k = 2,\ldots,n$, *the degree of nonholonomy equals* $n+2$.

2.6 Bound for the degree of nonholonomy

A first bound for this degree has been given by Laumond ([21]) as a direct consequence of the proof of controllability: we just have to remark that the vector fields (20) belong to $\mathcal{L}'_{2^{n+1}}(X_1, X_2)$. Thus the degree of nonholonomy is bounded by 2^{n+1}. However this bound is too large, as it can be seen in the examples with 1 or 2 trailers.

It has been proved in 1993 ([24,34]) that a better bound is the (n+3)-th Fibonacci number, which is defined by $F_0 = 0$, $F_1 = 1$, $F_{n+3} = F_{n+2} + F_{n+1}$. Luca and Risler have also proved that this bound is a maximum which is reached if and only if each trailer (except the last one) is perpendicular to the previous one.

Theorem 2.4. *The degree of nonholonomy* $\beta^n(q)$ *for the car with n trailers satisfies:*

$$\beta^n(q) \leq F_{n+3}.$$

Moreover, the equality happens if and only if $\theta_i - \theta_{i-1} = \pm\frac{\pi}{2}$, $i = 2,\ldots,n$.

Let us remark that this bound is exponential in n since the value of the n-th Fibonacci number is given by:

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right].$$

2.7 Form of the singular locus

The last problem is to determinate the form of the singular locus, which is given in Theorem 2.2. We already know the values of the degree of nonholonomy in two extremal cases:

- if no two consecutive trailers are perpendicular, $\beta^n(q) = n+2$;
- if each trailer is perpendicular to the previous one, $\beta^n(q) = F_{n+3}$.

We have now to characterize the states intermediate between these both cases.

For a given state q, we have the following sequence of dimensions:

$$2 = \dim L_1(X_1, X_2)(q) \leq \cdots \leq \dim L_k(X_1, X_2)(q) \leq \cdots \leq n+3. \tag{21}$$

Let us recall that, if this sequence stays the same in an open neighbourhood of q, the state q is a regular point of the control system; otherwise, q is a singular point (see §1.6). Thus to give the sequence (21) at any state q allows to characterize the singular locus.

To determinate the sequence (21), we only need the dimensions of the spaces $L_k(X_1, X_2)(q)$ such that $L_k(X_1, X_2)(q) \neq L_{k-1}(X_1, X_2)(q)$. For that we define, for $i \in \{1, n+3\}$:

$$\beta_i^n(q) = \min\{k \mid \dim L_k(X_1, X_2)(q) \geq i\}$$

In other words, the fact that $k = \beta_i^n(q)$ is equivalent to:

$$\begin{cases} \dim L_k(X_1, X_2)(q) & \geq i \\ \dim L_{k-1}(X_1, X_2)(q) & < i \end{cases} \tag{22}$$

The sequence (21) can be entirely deduced from the sequence $\beta_i^n(q)$, $i = 1, \ldots, n+3$. Hence the singular locus is completly characterized by the $\beta_i^n(q)$'s which we are going to study. Let us remark that $\beta_{n+3}^n(q)$ is the degree of non-holonomy $\beta^n(q)$.

According to its definition, $\beta_i^n(q)$ increases with respect to i, for i lesser than $\dim L(X_1, X_2)(q)$ (when i is strictly greater than this dimension, $\beta_i^n(q)$ is equal to $-\infty$). In fact we will establish (in Theorem 2.5) that this sequence is strictly increasing with respect to i for $2 \leq i \leq n+3$. In other words, we will prove that, for $2 \leq i \leq n+3$, $\beta_i^n(q) > -\infty$ and that $k = \beta_i^n(q)$ is equivalent to (compare with (22)):

$$\begin{cases} \dim L_k(X_1, X_2)(q) & = i \\ \dim L_{k-1}(X_1, X_2)(q) & = i-1 \end{cases}$$

We can also calculate easily the first values of these sequences. It is clear that the family X_1, X_2, $[X_1, X_2]$ is three dimensional for all q (see the examples $n = 1$ and 2). Then the dimensions of $L_1(X_1, X_2)(q)$ and $L_2(X_1, X_2)(q)$ are respectively 2 and 3 and we have, for all state q:

$$\beta_1^n(q) = 1 \; \beta_2^n(q) = 1 \; \beta_3^n(q) = 2. \tag{23}$$

Finally, for $q \in \mathbf{R}^2 \times (\mathcal{S}^1)^{n+1}$ and $1 \leq p < n$, we will denote the projection on the first $(n+3-p)$ coordinates of q by q^p, that is $q^p = (x, y, \theta_0, \ldots, \theta_{n-p})^T$. q^p belongs to $\mathbf{R}^2 \times (\mathcal{S}^1)^{n-p+1}$ and it can be seen as the state of a car with $n-p$ trailers. Hence we can associate to this state the sequence $\beta_j^{n-p}(q^p)$, $j = 1, \ldots, n-p+3$.

We can now give the complete characterization of the singular locus, *i.e.*, the computation of the $\beta_i^n(q)$ and the determination of a basis of $T_q(\mathbf{R}^2 \times (\mathcal{S}^1)^{n+1})$. The following theorem has been proved by F. Jean in ([19]). We restrict us to the case where the distances r_i equal 1.

Theorem 2.5. *Let a_p defined by $a_1 = \pi/2$ and $a_p = \arctan\sin a_{p-1}$. $\forall q \in \mathbf{R}^2 \times (\mathcal{S}^1)^{n+1}$, for $2 \leq i \leq n+3$, $\beta_i^n(q)$ is streactly increasing with respect to i and can be computed, for $i \in \{3, n+3\}$, by the following induction formulae:*

1. *if $\theta_n - \theta_{n-1} = \pm\frac{\pi}{2}$, then*

$$\beta_i^n(q) = \beta_{i-1}^{n-1}(q^1) + \beta_{i-2}^{n-2}(q^2)$$

2. *if $\exists p \in [1, n-2]$ and $\epsilon = \pm 1$ such that $\theta_k - \theta_{k-1} = \epsilon a_{k-p}$ for every $k \in \{p+1, n\}$, then*

$$\beta_i^n(q) = 2\beta_{i-1}^{n-1}(q^1) - \beta_{i-2}^{n-2}(q^2)$$

3. *otherwise,*

$$\beta_i^n(q) = \beta_{i-1}^{n-1}(q^1) + 1.$$

Moreover, at a point q, we can construct a basis $\mathcal{B} = \{B_i, i = 1 \ldots n+3\}$ of $T_q(\mathbf{R}^2 \times (\mathcal{S}^1)^{n+1})$ by:

$$\begin{cases} B_1 = X_1 \\ B_2 = X_2 \\ B_i = [X_1, \underbrace{X_2, \ldots, X_2}_{\beta_{i-1}^{n-1}(q^1)}, \underbrace{X_1, \ldots, X_1}_{\beta_i^n(q) - \beta_{i-1}^{n-1}(q^1) - 1}] \text{ for } i > 2, \end{cases}$$

where $[X_{i_1}, \ldots, X_{i_s}]$ denotes $[[\ldots[X_{i_1}, X_{i_2}], \ldots, X_{i_{s-1}}], X_{i_s}]$.

Let us consider the sequence $(\beta_i^n(q))_{i=2,\ldots,n+3}$ (we remove $\beta_1^n(q)$ because it is always equal to $\beta_2^n(q)$). For example, for $n = 2$, the sequence $(\beta_i^2(q))$ is equal to $(1, 2, 3, 5)$ on the hyperplanes $\theta_2 - \theta_1 = \pm\frac{\pi}{2}$. The complementary of these two hyperplanes are the regular points of the system and corresponds to the values $(1, 2, 3, 4)$ of the sequence $(\beta_i^2(q))$.

As we have seen in Theorem 2.2, the singular locus is the union of the the hyperplanes $\theta_k - \theta_{k-1} = \pm\frac{\pi}{2}$, $2 \leq k \leq n$. On each hyperplane we have a generic sequence $(\beta_i^n(q))$ and the non generic points are:

- either in the intersection with another hyperplane $\theta_j - \theta_{j-1} = \pm\frac{\pi}{2}$ which corresponds to the case *1* of Theorem 2.5;
- either in the intersection with an hyperplane $\theta_{k+1} - \theta_k = \pm a_2$, $(a_2 = \frac{\pi}{4})$ which corresponds to the case *2* of Theorem 2.5.

For these "more singular" sets, we have again some generic and some singular points that we can find with Theorem 2.5. We have then a stratification of the singular locus by the sequence $(\beta_i^n(q))$. Let us consider for instance the hyperplane $\theta_2 - \theta_1 = \frac{\pi}{2}$. The generic sequence $(\beta_i^n(q))$ is equal to $(1, 2, \ldots, n+1, n+3)$ (it is a direct application of the recursion formulae of Theorem 2.5). The non generic points are at the intersection with the hyperplanes $\theta_j - \theta_{j-1} = \pm\frac{\pi}{2}$, $j = 3, \ldots, n$ and with $\theta_3 - \theta_2 = \pm\frac{\pi}{4}$. On $\theta_2 - \theta_1 = \frac{\pi}{2}$, $\theta_3 - \theta_2 = \frac{\pi}{4}$, the generic sequence is $(1, 2, \ldots, n+1, n+4)$ and we can continue the decomposition.

Let us remark at last that Theorem 2.5 contains all the previous results. For instance, it proves that $\beta_{n+3}^n(q)$ is always definite (*i.e.*, $> -\infty$): the rank of $L(X_1, X_2)(q)$ at any point is then $n+3$ and the system is controllable. We can also compute directly the values of $\beta_{n+3}^n(q)$ and then its maximum, and so on.

3 Polynomial systems

3.1 Introduction

We will deal in this section with *polynomial systems*, *i.e.*, control systems in $\mathbf{R}^n$ made with vector fields $V_i = \sum_{j=1}^n P_i^j \partial X_i$, where the P_i^j's are polynomials in $X_1, \ldots, X_n$. Polynomial systems are important for "practical" (or "effective") purpose, because polynomials are the simplest class of functions for which symbolic computation can be used. Also, we can hope of global finiteness properties (on $\mathbf{R}^n$) for such systems, and more precisely of effective bounds in term of n and of a bound d on the degrees of the P_i^j.

In this section, we will study the degree of nonholonomy of an affine system without drift Σ made with polynomial vector fields $V_1, \ldots, V_s$ on $\mathbf{R}^n$, and prove that it is bounded by a function $\phi(n, d)$ depending only on the dimension n of the configuration space $\mathbf{R}^n$, and on a bound d on the degrees of the P_i^j. As a consequence, we have that the problem of controllability for a polynomial system $(V_1, \ldots, V_s)$ of degree $\leq d$ (with rational coefficients) is effectively decidable: take $x \in \mathbf{R}^n$, compute the value at x of the iterated brackets of $(V_1, \ldots, V_s)$ up to length $\phi(n, d)$. Then the system is controllable at x if and only if the vector space spaned by the values at x of these brackets is $\mathbf{R}^n$ (see above §1.4). For the controllability on $\mathbf{R}^n$, take a basis of $\mathcal{L}^{\phi(n,d)}$, *i.e.*, of the elements of degree $\leq \phi(n, d)$ of the Lie algebra $\mathcal{L}(V_1, \ldots, V_s)$. Then the system Σ is controllable on $\mathbf{R}^n$ if and only if this finite family of vector fields is of

rank n at any point $x \in \mathbf{R}^n$. But this is known to be effectively decidable: one has to decide if a matrix M with polynomial entries is of rank n at any point of $\mathbf{R}^n$. The matrix M is the matrix $(V_1, \ldots, V_s, V_{s+1}, \ldots, V_k)$, where the V_i's are the vector fields of Σ for $1 \leq i \leq s$, and for $s+1 \leq i \leq k$ a set of brackets of the form $[[\ldots[V_{i_1}, V_{i_2}], V_{i_3},]\ldots]V_{i_p}]$, with $1 \leq i_j \leq s$, spaning $\mathcal{L}^{\phi(n,d)}$. Let I be the ideal of $\mathbf{R}[X_1, \ldots, X_n]$ spaned by all the $n \times n$ minors of the matrix M. Then Σ is controllable on all $\mathbf{R}^n$ if and only if the zero set of I is empty, and that is effectively decidable (see for instance [15] or [17]).

The bound described here for the degree of nonholonomy is doubly exponential in n. A better bound (and in fact an optimal one) would be a bound simply exponential in n, *i.e.*, of the form $O(d^n)$ or $d^{O(n)}$, or again $d^{n^{O(1)}}$. For an optimal bound in a particular case, see Section 2 of this chapter, for the case of the car with n trailers. Note that this system is not polynomial.

3.2 Contact between an integral curve and an algebraic variety in dimension 2

In this section, we will work over the field $\mathbf{C}$, but all the results will be the same over the field $\mathbf{R}$. By the contact (or intersection multiplicity) between a smooth analytic curve γ going through the origin O in $\mathbf{C}^n$ and an analytic germ of hypersurface at O, $\{Q = 0\}$, we mean the order of $Q_{|\gamma}$ at O. More precisely, let $X_1(t), \ldots, X_n(t)$, $X_i(0) = 0$ be a parametrization of the curve γ near the origin ($X_i(t)$ are convergent power series in t). Then the contact of γ and $\{Q = 0\}$ at O is the order at 0 of the power series $Q(X_1(t), \ldots, X_n(t))$ (*i.e.*, the degree of the non zero monomial of lowest degree of this series). Let us give an example, for the convenience of the reader. Set $n = 2$, $Q(X, Y) = Y^2 - X^3$, $\gamma(t)$ defined by $X(t) = t^2 + 2t^5$, $Y(t) = t^3 + t^4$. We have $Q_{|\gamma} = (t^3 + t^4)^2 - (t^2 + 2t^5)^3 = 2t^7 +$ higher order terms; then the contact exponent between γ and the curve $\{Q = 0\}$ is 7.

Let us first recall some classical facts about intersection multiplicity. If $Q_1, \ldots, Q_p$ are analytic functions defined in a neighborhood of O, we will set $Z(Q_1, \ldots, Q_p)$ for the analytic germ at O defined by $Q_1 = \cdots = Q_p = 0$, and $\mathbf{C}\{X_1, \ldots, X_n\}$ for the ring of convergent power series.

Then, if in $\mathbf{C}^n$ we have $\{O\} = Z(Q_1, \ldots, Q_n)$, the intersection multiplicity at O of the analytic germ defined by $\{Q_i = 0\}$ $(1 \leq i \leq n)$ is by definition

$$\mu(Q_1, \ldots, Q_n) = \dim_{\mathbf{C}} \frac{\mathbf{C}\{X_1, \ldots, X_n\}}{(Q_1, \ldots, Q_n)} \tag{24}$$

Recall that the condition $\{O\} = Z(Q_1, \ldots, Q_n)$ (locally at O) is equivalent to the fact that the $\mathbf{C}$-vector space $\frac{\mathbf{C}\{X_1, \ldots, X_n\}}{(Q_1, \ldots, Q_n)}$ is of finite dimension. Recall

at last that when $Q_1, \dots, Q_n$ are polynomials of degrees $q_1, \dots, q_n$, we have $\mu(Q_1, \dots, Q_n) \leq q_1 \cdots q_n$ by Bézout's theorem, if $\dim_{\mathbf{C}} \frac{\mathbf{C}\{X_1,\dots,X_n\}}{(Q_1,\dots,Q_n)} < +\infty$.

Let $V = P_1\partial/\partial X_1 + \cdots + P_n\partial/\partial X_n$ be a polynomial vector field such that $V(O) \neq 0$, $\deg(P_i) \leq d$. Let $Q(X_1, \dots, X_n)$ be a polynomial of degree q. Set

$$\begin{aligned} Q_1 &= P_1\partial Q/\partial X_1 + \cdots + P_n\partial Q/\partial X_n \\ Q_2 &= P_1\partial Q_1/\partial X_1 + \cdots + P_n\partial Q_1/\partial X_n \\ &\vdots \\ Q_{n-1} &= P_1\partial Q_{n-2}/\partial X_1 + \cdots + P_n\partial Q_{n-2}/\partial X_n \end{aligned}$$

(*i.e.*, $Q_0 = Q$ and $Q_i =< P, \mathrm{grad} Q_{i-1} >= \sum_{j=1}^{n} P_j\, \partial Q_{i-1}/\partial X_j$, for $1 \leq i \leq n-1$); Q_1 is the Lie derivative of Q along the vector field V, and more generally, Q_i is the Lie derivative of Q_{i-1} along the vector field V.
We have the following:

Theorem 3.1. *Let V be a vector field in $\mathbf{C}^n$ whose coordinates are polynomials of degree $\leq d$, and such that $V(O) \neq 0$. Let γ be the integral curve of V going through O, and Q a polynomial of degree q. Assume $Q_{|\gamma} \not\equiv 0$, and that O is isolated in the algebraic set $Z(Q, Q_1, \dots, Q_{n-1})$ (which means that $\dim_{\mathbf{C}} \frac{\mathbf{C}\{X_1,\dots,X_n\}}{(Q,\dots,Q_{n-1})} < +\infty$). Then the contact exponent ν between Q and γ satisfies*

$$\nu \leq q q_1 \cdots q_{n-1} + n - 1, \tag{25}$$

where q_i is a bound for the degree of Q_i, namely $q_i = q + i(d-1)$.

Proof. We may assume $\nu \geq n$. Let $\gamma(t) : t \mapsto (X_1(t), \dots, X_n(t))$ be a smooth analytic parametrization of γ. By definition, ν is the order of the power series $Q \circ \gamma(t) = Q(X_1(t), \dots, X_n(t))$. Now, $Q_1 \circ \gamma(t) = Q_1(X_1(t), \dots, X_n(t))$ is the derivative of $Q \circ \gamma(t)$, and therefore is of order $\nu - 1$ at O. Similarly $Q_i \circ \gamma(t)$ is of order $\nu - i$ for $1 \leq i \leq n-1$. We have that the series $Q(X_1(t), \dots, X_n(t))$ is of the form $t^\nu v(t)$, *i.e.*, belongs to the ideal (t^ν) in $\mathbf{C}\{X_1, \dots, X_n\}$. Similarly, $Q_i(X_1(t), \dots, Q_n(t))$ belongs to the ideal $(t^{\nu-i})$.
Set γ^* for the ring homomorphism : $\mathbf{C}\{X_1, \dots, X_n\} \longrightarrow \mathbf{C}\{t\}$ induced by the parametrization of γ. The image of γ^* contains by assumption a power series of order one, *i.e.*, of the form $v(t) = tu(t)$, with $u(O) \neq 0$. Then the inverse function theorem implies that t itself is in the image of γ^*, *i.e.*, that γ^* is surjective. Hence we have a commutative diagramm of ring homomorphisms:

$$\begin{array}{ccc} \mathbf{C}\{X_1, \dots, X_n\} & \xrightarrow{\gamma^*} & \mathbf{C}\{t\} \\ \downarrow & & \downarrow \\ \frac{\mathbf{C}\{X_1,\dots,X_n\}}{(Q,Q_1,\dots Q_{n-1})} & \xrightarrow{\bar{\gamma}^*} & \frac{\mathbf{C}\{t\}}{(t^{\nu-n+1})} \end{array}$$

where the vertical arrows represent the canonical maps. Since γ^* is surjective, we have also that $\bar{\gamma}^*$ is surjective.

This implies that

$$\nu - n + 1 = \dim_{\mathbf{C}} \frac{\mathbf{C}\{t\}}{(t^{\nu-n+1})} \leq \dim_{\mathbf{C}} \frac{\mathbf{C}\{X_1, \ldots, X_n\}}{(Q, \ldots, Q_{n-1})} \leq qq_1 \cdots q_{n-1}$$

or $\nu \leq qq_1 \cdots q_{n-1} + n - 1$ as asserted, the last inequality coming from Bézout's theorem. ∎

Remark. One may conjecture that such a kind of result is valid (may be with a slightly different bound) without the hypothesis $\dim_{\mathbf{C}} \frac{\mathbf{C}\{X_1,\ldots,X_n\}}{(Q,\ldots,Q_{n-1})}$ finite. This would imply a simply exponential bound (*i.e.*, of the form $C(n)d^n$, or $d^{n^{O(1)}}$) for the degree of nonholonomy.

Notice that for Theorem 3.1, we may always assume that the polynomial $Q(X_1, \ldots, X_n)$ is *reduced* (or even *irreducible*), because if $Q = R_1 \ldots R_s$, the bound (25) for the R_i's implies the same bound for Q. In fact, it is enough to prove that if $Q = RS$, $r = \deg R$, $s = \deg S$, $q = r + s$, then

$$r(r+d-1)\cdots(r+(n-1)(d-1))+s(s+d-1)\cdots(s+(n-1)(d-1))+2(n-1) \leq q(q+d-1)\cdots(q+(n-1)(d-1))+n-1$$

which is immediate by induction on n.

If A is a $\mathbf{C}$-algebra, let us denote by $\dim A$ its dimension as a ring (it is its "Krull dimension"), and $\dim_{\mathbf{C}} A$ its dimension as a $\mathbf{C}$-vector space. If A is an analytic algebra, *i.e.*, $A = \frac{\mathbf{C}\{X_1,\ldots,X_n\}}{I}$ where I is an ideal, $I = (S_1, \ldots, S_q)$, then its dimension as a ring is the dimension (over $\mathbf{C}$) of the germ at O of the analytic space defined by $Z(S_1, \ldots, S_q)$. We have that $\dim_{\mathbf{C}} A < +\infty$ if and only if $\dim A = 0$.

Notice that Q_1 cannot be divisible by Q (since $Q \circ \gamma(t)$ is of order ν, and $Q_1 \circ \gamma(t)$ of order $\nu - 1$). Therefore, if Q is irreducible, we have

$$\dim \frac{\mathbf{C}\{X_1, \ldots, X_n\}}{(Q, Q_1)} = n - 2.$$

This implies that in Theorem 3.1, we may always assume that we have $\dim \frac{\mathbf{C}\{X_1,\ldots,X_n\}}{(Q,\ldots,Q_{n-1})} \leq n - 2$.

In particular, (25) is true in dimension 2 without additional hypothesis:

Corollary 3.2. *Let $V = P_1\partial/\partial X + P_2\partial/\partial Y$ be a polynomial vector field in the plane of degree $\leq d$, such that $V(O) \neq 0$, γ the integral curve of V by O, and $Q(X, Y)$ a polynomial of degree q such that $Q_{|\gamma} \not\equiv 0$. Then the contact exponent ν of Q and γ satisfies*

$$\nu \leq q(q + d - 1) + 1.$$

This corollary has first been proved by A. Gabrielov, J.-M. Lion and R. Moussu, [10].

3.3 The case of dimension n

We have the following result, due to Gabrielov ([12])

Theorem 3.3. *Let $V = \sum P_i \partial/\partial X_i$ be a polynomial vector field, with $P_i \in \mathbf{C}[X_1, \ldots, X_n]$ of degree $\leq d$, such that $V(O) \neq 0$, $Q(X_1, \ldots, X_n)$ a polynomial of degree $\leq q$ such that $Q_{|\gamma} \not\equiv 0$. Then the contact exponent ν between Q and γ satisfies*

$$\nu \leq 2^{2n-1} \sum_{k=1}^{n} [q + (k-1)(d-1)]^{2n} \tag{26}$$

Remark. This bound is polynomial in d and q and simply exponential in n. It is optimal (up to constants) since it comes from Example 2) below that there exists a lower bound also polynomial in d and q and simply exponential in n.

Remark. In 1988 Nesterenko ([27]) found a bound of the form

$$\nu \leq c(n) d^{n^2} q^n,$$

namely simply exponential in n when d is fixed, but doubly exponential in the general case.

Remark. In dimension 3, the following bound has been found by A. Gabrielov, F. Jean and J.-J. Risler, [9]:

$$\nu \leq q + 2q(q + d - 1)^2.$$

3.4 Bound for the degree of nonholonomy in the plane

In the two-dimensional case, we have the following bound for the degree of non-holonomy (see [29]):

Theorem 3.4. *Let $\Sigma = (V_1, \ldots, V_s)$ be a control system made with polynomial vector fields on $\mathbf{R}^2$ of degree $\leq d$; let $r(x)$ be the degree of nonholonomy of Σ at $x \in \mathbf{R}^2$. Then,*

$$r(x) \leq 6d^2 - 2d + 2 \tag{27}$$

Proof. Take $x = O$. Let as above (see §1.6) $L^i(O)$ be the vector space spaned by the values at O of the brackets of elements of Σ of length $\leq i$. We may assume $\dim L^1(O) = 1$, because otherwise the problem of computing $r(O)$ is trivial (if $\dim L^1(O) = 0$, then $L^s(O) = \{0\}$ $\forall s \geq 1$, and if $\dim L^1(O) = 2$, we have $L^1(O) = \mathbf{R}^2$ and $r(O) = 1$ by definition). We therefore assume that $V_1(O) \neq 0$, and set $V = V_1$. ∎

Lemma 3.5. *Assume $r(O) > 1$, which in our case implies that the system Σ is controllable at O. Then there exists $Y \in \Sigma$ such that* $\det(V, Y)_{|\gamma} \not\equiv 0$.

Proof. Assume that $\det(V, Y)_{|\gamma} \equiv 0$ $\forall Y \in \Sigma$. Then, in some neighborhood of O, any vector field $Y \in \Sigma$ is tangent to the integral curve γ of V from O. This implies that the system cannot be controllable at O, since in some neighborhood of O the accessible set from O would be contained in γ. ∎

Let us now state a Lemma in dimension n.

Lemma 3.6. *Let $V, Y_1, \ldots, Y_n$ be vector fields on $\mathbf{R}^n$. Then*

$$V.\det(Y_1, \ldots, Y_n) = \sum_{i=1}^{n} \det(Y_1, \ldots, [V, Y_i], \ldots, Y_n) + \operatorname{Div}(V).\det(Y_1, \ldots, Y_n).$$

Let us recall that $\operatorname{Div}(V) = \partial P_1/\partial X_1 + \partial P_2/\partial X_2 + \cdots + \partial P_n/\partial X_n$, *where* $V = P_1\partial/\partial X_1 + \cdots + P_n\partial/\partial X_n$.

Proof. This formula is classical. See for instance [13, Exercice page 93], or [26, Lemma 2.6]. ∎

of Theorem 3.4, continued. Let γ be the integral curve of V by O. By Lemma 3.5, there exists $Y \in \Sigma$ such that $\det(V, Y)_{|\gamma} \not\equiv 0$. Set $Q = \det(V, Y)_{|\gamma}$.

By Lemma 3.6, we have $V.\det(V, Y) = \det(V, [V, Y]) + \operatorname{Div} V \det(V, Y)$. Let ν be the order of contact of Q and γ. We have that $Q_{|\gamma} = a_\nu t^\nu + \cdots$, with $a_\nu \neq 0$, and that $(V.Q)_{|\gamma} = \nu a_\nu t^{\nu-1} + \cdots$ because $V_{|\gamma}$ can be identified with $\partial/\partial t$. Then $\det(V, [V, Y])_{|\gamma}$ is of order $\nu - 1$ in t, and we see that when differentiating ν times in relation to t, we find that

$$\det(V, [V[V, \ldots [V, Y] \ldots]])(O) \neq 0,$$

the bracket inside the parenthesis being of length $\nu+1$. This means by definition of $r(O)$ that $r(O) \leq \nu + 1$.

The polynomial Q is of degree $\leq 2d$, and V is a polynomial vector field of degree $\leq d$. Then Corollary 3.2 gives $\nu \leq 2d(2d + d - 1) + 1 = 6d^2 - 2d + 1$. ∎

Example. 1) Set

$$\Sigma \quad \begin{cases} V_1 = \partial/\partial X + X^d \partial/\partial Y \\ V_2 = Y^d \partial/\partial X \end{cases}$$

Then it should be easily seen that for this system, $r(O) = d^2 + 2d + 1$. The inequality $r(O) \geq d^2 + 2d + 1$ has been checked by F. Jean. This proves that the estimation (27) is asymptotically optimal in term of d, up to the constant 6.

2) Let in $\mathbf{R}^n$

$$\Sigma \quad \begin{cases} V_1 = \partial/\partial X_1 \\ V_2 = X_1^d \partial/\partial X_2 \\ \quad \vdots \\ V_n = X_{n-1}^d \partial/\partial X_n. \end{cases}$$

We see easily that for this system, $r(O) = d^{n-1}$, which means that in general $\phi(n, d)$ is at least exponential in n.

3.5 The general case

We have the following result, where for simplicity, and because it is the most important case, we assume the system controllable.

Theorem 3.7. *Let $n \geq 3$. With the above notation, let $r(x)$ be the degree of non-holonomy at $x \in \mathbf{R}^n$ for the control system Σ made with polynomial vector fields of degree $\leq d$. Let us assume that the system Σ is controllable. Then we have the following upper bound:*

$$r(x) \leq \phi(n, d), \quad \text{with } \phi(n, d) \leq 2^{n-2}(1 + 2^{2n(n-2)-2} d^{2n} \sum_{k=4}^{n+3} k^{2n}). \tag{28}$$

Proof. We first state without proof a result of Gabrielov [11].

Lemma 3.8. *Let $(V_1, \ldots, V_s)$ be a system of analytic vector fields controllable at O such that $V_1(O) \neq 0$. Let f be a germ of an analytic function such that $f(O) = 0$ and $f|_{\gamma(V_1)} \not\equiv 0$ ($\gamma(V_1)$ denotes the trajectory of V_1 going through O). Then there exists n vector fields $\chi_1, \ldots, \chi_n$ satisfying*

- *$\chi_1 = V_1$, χ_2 is one of the V_i, and, for $2 < k \leq n$, χ_k is either one of the V_i or belongs to the linear subspace generated by $[\chi_l, f\chi_m]$, for $l, m < k$;*
- *there exists a vector field $\chi_\epsilon = \chi_1 + \epsilon_2 \chi_2 + \cdots + \epsilon_{n-1}\chi_{n-1}$ such that*

$$\det(\chi_1, \ldots, \chi_n)|_{\gamma(\chi_\epsilon)} \not\equiv 0.$$

Let us assume $x = O$. For a generic linear function f, the conditions $f(O) = 0$ and $f|_{\gamma(V_1)} \not\equiv 0$ are ensured. We can then apply the lemma and obtain n vector fields $\chi_1, \ldots, \chi_n$. From the first point of Lemma 3.8, χ_k is a polynomial vector field of degree not exceeding $2^{k-2}d$. Thus the vector field χ_ϵ is polynomial of degree not exceeding $2^{n-3}d$ and the determinant $Q = \det(\chi_1, \ldots, \chi_n)$ is also polynomial. Its degree does not exceed $d + d + \cdots + 2^{n-2}d = 2^{n-1}d$.

The second point of Lemma 3.8 ensures that Q and χ_ϵ fulfill the conditions of Theorem 3.3. Then, applying (26), the contact exponent ν between Q and $\gamma(\chi_\epsilon)$ satisfies

$$\nu \leq 2^{2n^3-4n-1} \sum_{k=1}^{n} (4d + k - 1)^{2n}.$$

Each derivation of Q along χ_ϵ decreases this multiplicity by 1. Hence the result of ν consecutives derivations of Q along χ_ϵ does not vanish at O. By using Lemma 3.6, that means that there exists n brackets $\xi_k = [\chi_\epsilon, \ldots, [\chi_\epsilon, \chi_k] \ldots]$, with at most ν occurences of χ_ϵ, such that:

$$\det(\xi_1(O), \ldots, \xi_n(O)) \neq 0.$$

¿From the first point of Lemma 3.8, each χ_k is a linear combination with polynomial coefficient of brackets of the vector fields V_i of length not exceeding 2^{k-2}. This implies $\chi_k(O) \in L_{2^{k-2}}(\Sigma)(O)$ (this is the same reasoning as in the proof of Theorem 2.1). We have then $\chi_\epsilon(O) \in L_{2^{n-3}}(\Sigma)(O)$ and, $\forall k$, $\xi_k(O) \in L_{2^{n-2}+\nu 2^{n-3}}(\Sigma)(O)$.

Since $\det(\xi_1, \ldots, \xi_n) \not\equiv 0$, the subspace $L_{2^{n-2}+\nu 2^{n-3}}(\Sigma)(O)$ is of dimension n and then

$$r(O) \leq 2^{n-2}(1 + 2^{2n(n-2)-2} d^{2n} \sum_{k=4}^{n+3} k^{2n}).$$

■

References

1. A. A. Agrachev and R. V. Gamkrelidze, "The exponential representation of flows and the chronological calculus," *Mat. Sbornik* (N.S.), **107** (149), 467–532, 639, 1978. English transl.: *Math. USSR Sbornik*, **35**, 727–785, 1979.
2. A. Bellaïche, "The tangent space in sub-Riemannian Geometry," in *Sub-Riemannian Geometry*, A. Bellaïche and J.-J. Risler Ed., Progress in Mathematics, Birkhäuser, 1996.
3. A. Bellaïche, J.-P. Laumond and J. Jacobs, "Controllability of car-like robots and complexity of the motion planning problem," in *International Symposium on Intelligent Robotics*, 322–337, Bangalore, India, 1991.
4. A. Bellaïche, J.-P. Laumond and J.-J. Risler, "Nilpotent infinitesimal approximations to a control Lie algebra," in *Proceedings of the IFAC Nonlinear Control Systems Design Symposium*, 174–181, Bordeaux, France, June 1992.
5. R. W. Brockett, "Control theory and singular Riemannian geometry," in *New directions in applied mathematics*, P. J. Hilton and G. J. Young Ed., Springer-Verlag, 1982.
6. J. F. Canny, *The complexity of robot motion planning*, MIT Press, 1998.
7. W. L. Chow, "Über Systeme von linearen partiellen Differentialgleichungen erster Ordnung," *Math. Ann.*, **117**, 98–115, 1940.
8. M. Fliess, J. Levine, P. Martin and P. Rouchon, "On differential flat nonlinear systems," in *Proceedings of the IFAC Non linear Control Systems Design Symposium*, 408–412, Bordeaux, France, 1992.
9. A. Gabrielov, F. Jean and J.-J. Risler, *Multiplicity of Polynomials on Trajectories of Polynomials Vector Fields in C^3*, Preprint, Institut de Mathématiques, Univ. Paris VI, 1997.
10. A. Gabrielov, J.-M. Lion and R. Moussu, "Ordre de contact de courbes intégrales du plan," *CR Acad. Sci. Paris*, **319**, 219–221, 1994.
11. A. Gabrielov, "Multiplicities of Zeroes of Polynomials on Trajectories of Polynomial Vector Fields and Bounds on Degree of Nonholonomy," *Mathematical Research Letters*, **2**, 437–451, 1995.
12. A. Gabrielov, *Multiplicity of a zero of an analytic function on a trajectory of a vector field*, Preprint, Purdue University, March 1997.
13. C. Godbillon, *Géométrie différentielle et Mécanique analytique*, Hermann, Paris, 1969.
14. N. Goodman, *Nilpotent Lie groups*, Springer Lecture Notes in Mathematics, **562**, 1976.
15. S. Grigoriev, "Complexity of deciding Tarski algebra," *J. Symb. Comp.*, **5**, 37–64, 1988.
16. M. Gromov, "Carnot-Carathéodory spaces seen from within," in *Sub-Riemannian Geometry*, A. Bellaïche and J.-J. Risler Ed., Progress in Mathematics, Birkhäuser, 1996.
17. J. Heinz, M.-F. Roy and P. Solerno, "Sur la complexité du principe de Tarski-Seidenberg," *Bull. Soc. Math. Fr.*, **118**, 101–126, 1990.
18. H. Hermes, "Nilpotent and high-order approximations of vector field systems," *SIAM Review*, **33** (2), 238–264.

19. F. Jean, "The car with N trailers: characterization of the singular configurations," *ESAIM: COCV, http://www.emath.fr/cocv/*, **1**, 241–266, 1996.
20. J.-P. Laumond, "Singularities and Topological Aspects in Nonholonomic Motion Planning," in *Nonholonomic Motion Planning*, 149–199, Z. Li and J. Canny Ed., Klumer Academic Publishers, 1993.
21. J.-P. Laumond, "Controllability of a multibody mobile robot," in *Proceedings of the International Conference on Advanced robotics and Automation*, 1033–1038, Pisa, Italy, 1991.
22. J-P. Laumond, P. E. Jacobs, M. Taix and R. M. Murray, "A motion planner for nonholonomic mobile robot," in *LAAS-CNRS Report*, oct. 1992.
23. J.-P. Laumond and J.-J. Risler, "Nonholonomics systems: Controllability and complexity," *Theoretical Computer Science*, **157**, 101–114, 1996.
24. F. Luca and J.-J. Risler, "The maximum of the degree of nonholonomy for the car with n trailers," *4th IFAC Symposium on Robot Control*, 165–170, Capri, 1994.
25. R. M. Murray and S. S. Sastry, "Nonholonomic Motion Planning: Steering using sinusoids," *IEEE Transactions on Automatic Control*, **38** (5), 700–716, May 1993.
26. A. Nagel, E. M. Stein and S. Wainger, "Metrics defined by vector fields," *Acta Math.*, **155**, 103–147, 1985.
27. Y. V. Nesterenko, "Estimates for the number of zeros of certain functions," *New Advances in transcendance Theor*, 263–269, A. Baker Ed., Cambridge, 1988.
28. P. K. Rashevsky, "Any two points of a totally nonholonomic space may be connected by an admissible line," *Uch. Zap. Ped. Inst. im. Liebknechta, Ser. Phys. Math.*, **2**, 83–94, 1938 (Russian).
29. J.-J. Risler, "A bound for the degree of nonholonomy in the plane," *Theoretical Computer Science*, **157**, 129–136, 1996.
30. J. T. Schwartz and M. Sharir, "On the 'Piano Movers' problem II: general techniques for computing topological properties of real algebraic manifolds," *Advances in Applied Mathematics*, **4**, 298–351, 1983.
31. A. Seidenberg, "On the length of Hilbert ascending chain," *Proc. Amer. Math. Soc.*, **29**, 443–450, 1971.
32. E. Sontag, "Some complexity questions regarding controllability," *Proc. of the 27-th IEEE Conf. on Decision and Control*, 1326–1329, Austin, 1988.
33. O. J. Sørdalen, "Conversion of the kinematics of a car with n trailers into a chain form," *IEEE International Conference on Robotics and Automation*, 1993.
34. O. J. Sørdalen, "On the global degree of non holonomy of a car with n trailers," 4th IFAC Symposium on Robot Control, 343–348, Capri, 1994.
35. P. Stefan, "Accessible sets, orbits, and foliations with singularities," *Proc. London Math. Soc.*, **29** (3), 699–713, 1974.
36. H. J. Sussmann, "Orbits of families of vector fields and integrability of distributions," *Trans. Amer. Math. Soc.*, **180**, 171–188, 1973.
37. H. J. Sussmann, "Lie brackets, Real Analyticity and Geometric control," in *Differential Geometric Control Theory*, Brockett, Millman, Sussmann Ed., Birkäuser, 1–116, 1982.

Optimal Trajectories for Nonholonomic Mobile Robots

P. Souères[1] and J.-D. Boissonnat[2]

[1] LAAS - CNRS, Toulouse
[2] INRIA, Sophia Antipolis

1 Introduction

From a kinematic point of view, the main characteristic of wheeled robots is the nonholonomic rolling without slipping constraint of the wheels on the floor, which forces the vehicle to move tangentially to its main axis. This reduction of the set of accessible velocities at each time makes the path planning problem particularly difficult. Among the different methods devoted to solve this problem we want to focus on those based on the characterization of shortest paths or time-optimal paths, which turn out to be particularly efficient. Indeed, the knowledge of an optimal strategy for linking any two configurations allows to determine simple canonical paths and provides a topological modeling of the problem by defining a new "distance function" taking into account the nonholonomic nature of the system. Unfortunately, the characterization of optimal paths for this class of nonlinear systems is not an easy task.

The works presented in this chapter are based on Pontryagin's Maximum Principle (PMP) which constitutes a generalization of Lagrange's problem of the calculus of variation. PMP is a local reasoning based on the comparison of trajectories corresponding to infinitesimally close control laws. It provides necessary conditions for paths to be optimal. Nevertheless, though this condition brings a very strong information about the nature of optimal paths for certain kind of systems, it turns out to be insufficient to solve the optimal control problems we are interested on.

Indeed, on the one hand the nonlinear nature of these systems makes the adjoint differential equations seldom integrable. Therefore, in the most part of cases, the necessary condition of PMP only provides a local characterization of optimal trajectories. On the other hand, the study of such systems has shown that the set of accessible configurations at each time, is neither smooth nor convex. More precisely, it appears that the boundary of this set is made up by several smooth pieces corresponding to the propagation of several wave fronts. This is due at one and the same time to the difficulty of moving sideways and the natural symmetries of the problem.

For these latter reasons, the local nature of PMP cannot provide a tool to compare the cost of trajectories corresponding to different wave fronts. Therefore, this local information needs to be completed with a global study.

By combining these two approaches it is sometimes possible to get a better characterization of the solution. In this way, a first interesting result is the determination of a sufficient family of trajectories i.e. a family of trajectories containing an optimal solution for linking any two configurations. Whenever this family is small enough and sufficiently well specified it is possible to compare the cost of trajectories by means of a numerical method. Nevertheless, the ultimate goal one wants to reach is to achieve the determination of an optimal control law for steering the representative point from any point of the phase space to a given target set, i.e. to solve the synthesis problem.

Four works are presented in this chapter, devoted to the search of optimal paths for various models of wheeled robots. These problems are stated in the free phase space i.e. without obstacles. We have been able to solve the synthesis problem for only two of these models. The two other works provide an incomplete characterization of optimal solutions, bringing to the fore various kind of difficulties that can be encountered in studying such problems.

The paper is organized as follows: The models of wheeled robots and their related optimization problems are presented in section 2. The third section constitutes a survey of the definitions and results from optimal control theory which have been useful for these different works: Fillipov's existence theorem, Pontryagin's Maximum Principle (PMP) and Boltianskii's sufficient optimality condition. In particular, we give a geometric description of PMP in order to point out the local nature of this reasoning. The last four sections present successively the works related to each model.

2 Models and optimization problems

The aim of the works presented in this chapter is to characterize optimal trajectories verifying the nonholonomic constraints of mobile robots. Therefore, in order to get the simplest expression of the problem, we consider mathematical models defined upon the kinematic constraints inherent in these wheeled vehicles, without taking into account their dynamics. Classically, these models are described by differential autonomous[1] systems such as:

$$\frac{dx^i}{dt} = f^i(x^1, x^2, \ldots, x^n, u^1, u^2, \ldots, u^m) \tag{1}$$

[1] The function f does not depend explicitly on time.

where the x^i characterize the robot's coordinates in the phase space and the control parameters u^i express the existence of "rudders" such as the steering wheel or the brake-accelerator function. Once the control parameters are defined as time-varying functions $u_j = u_j(t), j = 1, \ldots, m$, the corresponding trajectory solution of (1) is uniquely determined by the choice of initial conditions $x^i(t_0) = x_0^i, i = 1, \ldots, n$.

2.1 Dubins' and Reeds-Shepp's car

The model of a car-like robot considered here, describes the two principal kinematic constraints of an usual car. The first one is the rolling without slipping constraint which obliges the vehicle to move tangentially to its main axis. The second constraint is due to the bound on the turning wheels' angle and prevents the car from moving on trajectories whose radius of curvature is lower than a given threshold R. A configuration of the car is represented by a triple $(x, y, \theta) \in \mathbf{R}^2 \times S^1$, where (x, y) are the coordinates of a reference point of the robot with respect to a Cartesian frame, and θ is the angle between the positive x-axis and the robot's main axis, see figure (1). With this modeling, the rolling without slipping constraint is expressed by the following equation:

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0$$

For our purpose, the direction of front wheels is not relevant, we only need to consider that the bound on the angle of steer ϕ induces an upper bound on the trajectories' curvature.

Therefore, the kinematics of the vehicle is described by the differential system (2) involving two control parameters u_1 and u_2 which represent respectively the algebraic value of the linear and angular velocities.

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{R} \end{pmatrix} u_2 \qquad (2)$$

This kinematic model was introduced by Dubins in 1957 [16] who set the problem of characterizing the shortest paths for a particle moving forward in the plane with a constant linear velocity ($u_1 = k$). Later, Reeds and Shepp [31] considered the same problem, when backwards motions are allowed ($|u_1| = k$). In both cases, as the modulus of the linear velocity keeps constant, the shortest path problem is equivalent to the minimum-time problem.

In the sequel without lost of generality we will fix the value of the constants: $R = 1$ and $k = 1$.

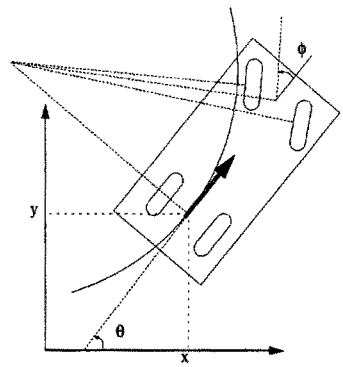


Fig. 1. The car-like model

2.2 Dubins' car with inertial angular velocity

As we will see later in detail, the optimal solutions of Dubins' problem are sequences of line segments and arcs of circle of minimal radius. Therefore, the curvature along the trajectory does not vary continuously. As a consequence, any real robot following such a trajectory would be constrained to stop at each curvature discontinuity.

In order to avoid this problem, Boissonnat, Cerezo and Leblond [3] have proposed a dynamic extension of Dubins' problem by controlling the angular acceleration of the car instead of its angular velocity. The curvature κ is now considered as a new phase variable, and the angular acceleration v is bounded inside a compact interval $[-B, B]$.

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\kappa} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ \kappa \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} v \tag{3}$$

For this problem, as for Dubins problem, minimizing the time comes to the same as minimizing the length.

2.3 The robot HILARE

The locomotion system of Hilare the robot of LAAS-CNRS consists of two parallel independently driven wheels and four slave castors. The velocities v_r and v_l of the right and left driven wheels are considered as phase variables and

a configuration of the robot is a 5-uple (x, y, θ, v_r, v_l). The accelerations a_r and a_l of each driven wheel are the inputs to the following control system:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_r \\ \dot{v}_l \end{pmatrix} = \begin{pmatrix} \frac{v_r+v_l}{2}\cos\theta \\ \frac{v_r+v_l}{2}\sin\theta \\ \frac{v_r-v_l}{d} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} a_r + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} a_l \tag{4}$$

where $a_r, a_l \in [-a_{\max}, a_{\max}]$, and $d > 0$ is the distance between the wheels. In this case the trajectories' curvature is not bounded and the robot can turn about its reference point.

For this model, we consider the problem of characterizing minimum-time trajectories linking any pair of configurations where the robot is at rest i.e verifying $v_r = v_l = 0$.

3 Some results from Optimal Control Theory

3.1 Definitions

Let us now define the notion of dynamical system in a more precise way. Let M be a n-dimensional manifold, and U a subspace of $\mathbf{R}^m$. We study the motion of a representative point $x(t) = (x^1(t), \ldots, x^n(t))$ in the phase space M, depending on the control law $u(t) = (u^1(t), \ldots, u^m(t))$ taking its values in the control set U. In this chapter, we define the set of admissible control laws as the class of measurable functions from the real time interval $[t_0, t_1]$ to U. As we said in the previous section, the motion of the representative point is described by an autonomous differential system of the form:

$$\frac{dx^i}{dt} = f^i(x(t), u(t)) \quad i = 1, \ldots, n \tag{5}$$

We consider now a function $L(x, u)$ defined on the product $M \times U$, continuously differentiable with respect to its arguments. Given any two points x_0 and x_1 in the phase space M, we want to characterize, among all the control laws steering the representative point from x^0 to x^1, one (if exists) minimizing the functional:

$$J = \int_{t_0}^{t_1} L(x(\tau, u), u(\tau))d\tau$$

Remark 1. *The initial and final time t_0 and t_1 are not fixed* a priori, *they depend on the control law $u(t)$.*

Definition 1. *Two trajectories are said to be* equivalent *for transferring the representative point from x_0 to x_1 if they their respective costs are equal.*

In the sequel we restrict our study to the minimum-time problem. In this case $L(x,u) \equiv 1$ and $J = t_1 - t_0$.

In chapter 1 it has been shown that the models described in the previous section are fully controllable in their phase space M, i.e. given any two configurations x_0 and x_1 in M there always exists a trajectory, solution of (5), linking x_0 to x_1. Nevertheless it is not possible to deduce from this result whether a minimum-time feasible path from x_0 to x_1 exists or not. This last question constitutes an important field of interest of optimal control theory (cf [13] for a detailed survey). In particular, there exist some general theorems due to Fillipov, ensuring the existence of optimal paths under some convexity hypotheses. The next subsection presents two theorems that will be sufficient for our purpose.

3.2 Existence of optimal paths

Let M be an open subset of $\mathbf{R}^n$ or a n-dimensional smooth manifold, and U a subset of $\mathbf{R}^m$.

Theorem 1. (Fillipov's general theorem for minimum-time problems)
Let x_0, x_1 be two points in M. Under the following hypotheses there exists a time-optimal trajectory solution of (5) linking x_0 to x_1.

H_1 - f is a continuous function of t, u, x and a continuously differentiable function of x.

H_2 - the control set U is a **compact** subset of $\mathbf{R}^m$. Furthermore, when u varies in U, the image set F(t,x) described by $f(x(t),u(t))$ is **convex** for all $t, x \in [t_0,t_1] \times M$.

H_3 - there exists a constant C such that for all $(t,x) \in [t_0,t_1] \times M$:

$$< x, f(t,x,u) > \leq C\,(1 + |x|^2)$$

H_4 - there exists an admissible trajectory from x_0 to x_1

Remark 2. - *The hypothesis H_3 prevents from a finite escape time of the phase variable x for any admissible control law $u(.)$.*

When f is a linear function of the control parameters u_i of the form:

$$f(x,u) = g_1(x)\,u_1 + \ldots + g_m(x)\,u_m, \tag{6}$$

there exists a simpler version of this result given by the next theorem.

Theorem 2. Let x_0 and x_1 be two points in M. Under the following hypotheses there exists an optimal trajectory solution of (6) linking x_0 to x_1.

H_1 - the g_i are locally lipschitzian functions of x.
H_2 - the control set U is a **compact convex** subset of $\mathbf{R}^m$
H_3 - there exists an admissible trajectory from x_0 to x_1
H_4 - The system is complete, in the sense that given any point $x_0 \in M$, and any admissible control law $u(.)$, there exists a corresponding trajectory $x(t,u)$ defined on the whole time interval $[t_0, t_1]$ and verifying $x(t_0, u) = x_0$.

3.3 Necessary conditions: Pontryagin's Maximum Principle

Pontryagin's Maximum Principle (PMP) provides a method for studying variational problems in a more general way than the classical calculus of variation does. Indeed, when the control set U is a closed subset of $\mathbf{R}^m$, the Weierstrass' condition characterizing the minimum of the cost functional is no more valid. The case of closed control set is yet the most interesting one for modeling concrete optimal control problems. PMP provides a necessary condition for the solutions of a general control systems to be optimal for various kind of cost functional. In this chapter we restrict our statement of PMP to minimum-time problems.

We consider the dynamical system (5) where x belongs to an open subset $\Omega \subset \mathbf{R}^n$ or a smooth n-dimensional manifold M.

Definition 2.

- Let ψ be a n-dimensional real vector, we define the Hamiltonian *of system (5) as the $\mathbf{R}$-valued function H defined on the set $\mathbf{R}^n_* \times \Omega \times U$ by:*

$$H(\psi, x, u) = < \psi, f(x,u) > \tag{7}$$

where $\mathbf{R}^n_* = \mathbf{R}^n \setminus \{0\}$, and $< .,. >$ is the usual inner product of $\mathbf{R}^n$.

- If $u(.) : [t_0, t_1] \to U$ is an admissible control law and $x(t) : [t_0, t_1] \to \Omega$ the corresponding trajectory, we define the adjoint vector *for the pair (x,u) as the absolutely continuous vector function ψ defined on $[t_0, t_1]$, taking its values in $\mathbf{R}^n_*$ which verifies the following* adjoint equation *at each time $t \in [t_0, t_1]$:*

$$\dot{\psi}(t) = -\frac{\partial H}{\partial x}(\psi(t), x(t), u(t)) \tag{8}$$

Remark 3. *As H is a linear function of ψ, $\dot{\psi}$ is also a linear function of ψ. Therefore, either $\psi(t) \neq 0 \; \forall t \in [t_0, t_1]$, or $\psi(t) \equiv 0$ on the whole interval $[t_0, t_1]$; in the latter case, the vector ψ is said to be* trivial.

Theorem 3. (Pontryagin's Maximum Principle) Let $u(.)$ be an admissible control law defined on the closed interval $[t_0, t_1]$ and $x(t)$ the corresponding trajectory. A necessary condition for $x(t)$ to be time-optimal is that there exists an absolutely continuous non trivial adjoint vector $\psi(t)$ associated to the pair (x, y), and a constant $\psi_0 \leq 0$ such that $\forall t \in [t_0, t_1]$:

$$H(\psi(t), x(t), u(t)) = \max_{v \in U}(H(\psi(t), x(t), v(t)) = -\psi_0 \tag{9}$$

Definition 3.

- A control law $u(t)$ satisfying the necessary condition of PMP is called an extremal *control law. Let $x(t, u)$ be the corresponding trajectory and $\psi(t)$ the adjoint vector corresponding to the pair (x, u); the triple (x, u, ψ) is also called* extremal.

- To study the variations of the Hamiltonian $H = \Sigma_i \dot{x}_i(t)\ \psi_i(t)$ one can rewrite H in the form: $H = \Sigma_i \phi_i(t)\ u_i(t)$ where the ϕ_i, called switching functions *determine the sign changes of u_i.*

Sometimes the maximization of the function H does not define a unique control law. In that case the corresponding control is called *singular*:

Definition 4. *A control $u(t)$ is* singular *if there exists a nonempty subset $W \subset U$ and a non empty interval $I \subset [t_0, t_1]$ such that $\forall t \in I$, $\forall w(t) \in W$:*

$$H(\psi(t), x(t), u(t)) = H(\psi(t), x(t), w(t))$$

In particular, when a switching function vanishes over a non empty open interval of time, the corresponding control law comes singular. In that case, all the derivatives of the switching function also vanish on this time interval, providing a sequence of equations. From these relations, it is sometimes possible to characterize the value of the corresponding singular control.

The following theorem allows to compute easily those derivatives in terms of Lie brackets.

Theorem 4. Let Z be a smooth vector field defined on the manifold M and (x, u, ψ) an extremal triple for the system (6).

The derivative of the function $\phi_Z(.) : t \longrightarrow < \psi(t), Z(x(t)) >$ is defined by:

$$\dot{\phi}_Z(t) = \sum_{i=1}^{r} < \psi, [g_i, Z]x(t) > .u_i$$

Let us now define the notion of reachable set:

Definition 5. - *We denote by* $\mathcal{R}(x_0, T)$, *and we call* set of accessibility from x_0 in time T, *the set of points* $x \in M$ *such that there exists a trajectory solution of (5) transferring the representative point from* x_0 *to* x *in a time* $t \leq T$.

- *The set* $\mathcal{R}(x_0) = \bigcup_{0 \leq T \leq \infty} \mathcal{R}(x_0, T)$ *is called* set of accessibility from x_0.

When the system is fully controllable (controllable from any point) the set of accessibility $\mathcal{R}(x_0)$ from any point x_0 is equal to the whole manifold M. Otherwise, $\mathcal{R}(x_0)$ is restricted to a closed subset of M. In this latter case there exists another version of PMP.

Theorem 5. (PMP for boundary trajectories) Let $u(.) : [t_0, t_1] \rightarrow U$ be an admissible control law, and $x(t)$ the corresponding trajectory transferring the representative point from x_0 to a point x_1 belonging to the boundary $\partial\mathcal{R}(x_0)$ of the set $\mathcal{R}(x_0)$. A necessary condition for the trajectory $x(t, u)$ to be time-optimal, is that there exists a non-trivial adjoint vector ψ verifying relation (9) with $\psi_0 = 0$

Definition 6. *An extremal triple* (x, u, ψ) *such that* $\psi_0 = 0$ *is called* abnormal

Commentary about PMP: It is often difficult to extract a precise information from PMP. Indeed, it is seldom possible to integrate the adjoint equations or to characterize the singular control laws. Furthermore, one can never be sure to have got all the information it was possible to deduce from PMP. Sometimes, the information obtained is very poor, and the set of potential solutions too large.

An interesting expected result is the characterization of a *sufficient family* of trajectory i.e. a family of trajectory containing an optimal solution for linking any pair of points $(x_0, x_1) \in M$. When this family is small enough, and sufficiently well specified the optimal path may be selected by means of a numerical test.

Nevertheless, the ultimate goal one wants to reach is the exact characterization of the optimal control law allowing to steer the point between any two states of M. However, though it is possible to deduce directly the structure of minimum-time trajectories from PMP for linear systems, the local information is generally insufficient to conclude the study in the case of nonlinear systems. As we will see in the sequel, it is yet sometimes possible to complete the local information provided by PMP by making a geometric study of the problem. When the characterization of optimal path is complete, a synthetic way of representing the solution is to describe a network of optimal paths linking any point of the state space to a given target point. The following definition due to Pontryagin states this concept in a more precise way.

Definition 7. *For a given optimization problem, we call* synthesis function, *a function $v(x)$ (if it exists) defined in the phase space M and taking its values in the control set U, such that the solutions of the equation:*

$$\frac{dx}{dt} = f(x, v(x))$$

are optimal trajectories linking any point of M to the origin. The problem of characterizing a synthesis function is called synthesis problem *and the corresponding network of optimal paths is called a* synthesis of optimal paths *on M.*

A geometric illustration of PMP: In the statement of PMP, optimal control laws are specified by the maximization of the inner product of two vectors. In the rest of this section, drawing our inspiration from a work by H. Halkin [21], we try to give a geometric interpretation of this idea by pointing out the analogy between optimal control and propagation phenomena. In this, we want to focus on the local character of PMP in order to point out its insufficiency for achieving the characterization of shortest paths for nonholonomic problems. This remark emphasizes the necessity to complete the local reasoning by making use of global arguments.

At the basis of the mathematical theory of optimal process stands the *principle of optimal evolution* which can be stated as follows:

" If $x(t,u)$ is an optimal trajectory starting from x_0 at time t_0, then at each time $t \geq t_0$ the representative point must belong to the boundary $\partial\mathcal{R}(x_0,t)$ of the set $\mathcal{R}(x_0,t)$"

For some physical propagation phenomena, such as the isotropic propagation of a punctual perturbation on the surface of water, the wavefront associated with the propagation coincides at each time with the boundary of the set of accessibility. Let us consider first the simple propagation of a signal starting at a point x_0 such that the set of accessibility $\mathcal{R}(x_0,t)$ at each time t be smooth and convex with a unique tangent hyperplane defined at each boundary point.

As in geometrical optics, at each time t and at each boundary point x, we can define the wavefront velocity as a nonzero vector $\boldsymbol{V}(x,t) = V(x,t)\ \boldsymbol{k}(x,t)$ where $V(x,t)$ is a $\mathbf{R}$-valued function of x and t, and $\boldsymbol{k}(x,t)$ a unit vector outward normal to the hyperplane tangent to $\mathcal{R}(x_0,t)$ at x. Now, according to the principle of optimal evolution, if $x(t,u)$ is an optimal trajectory starting at x_0, the following two conditions must be verified:

- For any admissible motion, corresponding to a control $w(t)$, the projection of the representative point velocity $\dot{x}(t,w) = f(x,w)$ on the line passing through $x(t,w)$ and whose direction is given by the vector $k(x,t)$, is at most equal to the wavefront speed $V(x,t)$.

$$< f(x(t), w(t)), k(x,t) > \leq V(x,t) \tag{10}$$

- With the optimal control u, the representative point must keep up with the wavefront $\partial\mathcal{R}(x_0,t)$ i.e. the projection of the representative point's velocity on the normal vector $k(x,t)$ determines the wavefront velocity.

$$< f(x(t), u(t)), k(x,t) > = V(x,t) \tag{11}$$

Now, by identifying the adjoint vector $\psi(x,t)$ with $V(x,t)$ we can make a link between relations (10, 11) and the maximization of the Hamiltonian defined by (9).

Though this analogy with propagation phenomena provides a good geometric meaning of this principle of optimization, it is not easy to generalize this idea to any dynamical system. Indeed, for a general system, the set $\mathcal{R}(x_0,t)$ is not necessarily convex and its boundaries are not necessarily smooth. In order to get a geometric meaning of Pontryagin's result in the general case, it is convenient to consider the cost functional J as a new phase variable x^0, and to manage our reasoning in the augmented phase space $\mathbf{R} \times \Omega \subset \mathbf{R}^{n+1}$. Therefore, at each time, the velocity vector of the representative point $X = (x^0, x) = (x^0, x^1, \ldots .x^n)$ corresponding to the control law $u(.)$ is given by $\hat{f}(X,u) = (L(x,u), f^1(x,u), \ldots, f^n(x,u))$. With this representation the optimization problem becomes:

"Let $\mathcal{D}$ be the line passing through $(0,x_1)$ parallel to the x^0-axis. Among all the trajectories starting at $X_0 = (0,x_0)$ and reaching $\mathcal{D}$, find one, if exists, which minimizes the first coordinate x^0 of the point of intersection $X_1 = (x^0, x_1)$ with $\mathcal{D}$."

As before we define the set of accessibility $\mathcal{R}(X_0,t)$ from X_0 in the augmented phase space. Now, it is easy to prove that any optimal path must verify the principle of optimality. Indeed, if the point X_1 of $\mathcal{D}$, reached at time t_1 with control u, lies in the interior of $\mathcal{R}(X_0,t_1)$, there exists necessarily a neighbourhood of X_1 containing a point of $\mathcal{D}$ located "under" X_1 and the control u cannot be optimal. Furthermore, due to the smoothness properties of the function $\hat{f}$, if the point X, reached at time $\tau \in [t_0,t_1]$ with u, is in the interior of $\mathcal{R}(X_0,\tau)$, then for all $t \geq \tau$ the representative point will belong to the interior of $\mathcal{R}(X_0,t)$.

Now, let $X(t,u)$ be a trajectory starting at X_0, optimal for reaching the line $\mathcal{D}$. In order to use the same reasoning as before, Pontryagin's *et al* have proven that it is still possible to construct a separating hyperplane by using the following idea: By replacing $u(.)$ by other admissible control laws on "small" time intervals they define new admissible control laws $\tilde{u}$ infinitesimaly close to

u. Then, a part of their proof consists in showing that the set of point $X(t,\tilde{u})$ reached at each time t by the "perturbed" trajectories constitutes a convex cone $\mathcal{C}$ with vertex $X(t,u)$, contained in $\mathcal{R}(X_0,t)$. This cone locally approximates the set $\mathcal{R}(X_0,t)$ and does not contain the half-line $\mathcal{D}^-$ starting at $X(t,u)$ in the direction of the decreasing x^0. It is then possible to find an hyperplane $\mathcal{H}$ tangent to $\mathcal{C}$ at $X(t,u)$, separating $\mathcal{C}$ and the half-line $\mathcal{D}^-$, and containing the vector $\hat{f}(t,u)$. Now, the reasoning is the same as before; the adjoint vector $\hat{\psi} = (\psi^0, \psi^1, \ldots, \psi^n)$ is defined, up to a multiplicative constant, as the vector outward orthogonal to this hyperplane at each time. Following the principle of optimal evolution, the projection of the vector $\hat{f}(t,u)$ on the line parallel to $\hat{\psi}(t)$, passing through $x(t,u)$, must be maximal for the control u.

The case of nonholonomic systems: As we will see later, the nonholonomic rolling without slipping constraint, characteristic of wheeled robots, makes their displacement anisotropic. Indeed, although forwards motion can be easily performed, moving sideways may require numerous manoeuvres. For this reason, and due to the symmetry properties of such systems, the set of accessibility (in time) is generally not convex and its boundary does not coincide everywhere with the wavefront associated with the propagation. Instead of this, we will show later that the boundary is made up by the propagation of several intersecting wavefronts. Therefore, a local method like PMP, based on the comparison of very close control laws cannot be a sufficient tool to compare the cost of trajectories corresponding to different wavefronts. This very important point will be illustrated in section 4 through the construction of a synthesis of optimal paths for the Reeds-Shepp car.

So far, we only have stated necessary conditions for trajectories to be optimal. We now present a theorem by V. Boltyanskii which states sufficient optimality conditions under very strong hypotheses.

3.4 Boltyanskii's sufficient conditions

In this section we recall Boltianskii's definition of a regular synthesis as stated in [4]. This concept is based on the definition of a *piecewise-smooth set.*

Let M be a n-dimensional vector space, and Ω an open subset of M. Let E an s-dimensional vector space ($s \leq n$) and $K \subset E$ a bounded, s-dimensional convex polyhedron. Assume that in a certain open set of E containing K are given n continuously differentiable functions $\varphi^i(\xi^1, \xi^2, \ldots, \xi^s)$, $(i = 1, 2, \ldots, n)$ such that the rank of the matrix of partial derivatives $(\frac{\partial \varphi^i}{\partial \xi^j})$, $(i = 1, 2, \ldots, n)$, $(j = 1, 2, \ldots, s)$ be equal to s at every point $\xi \in K$.

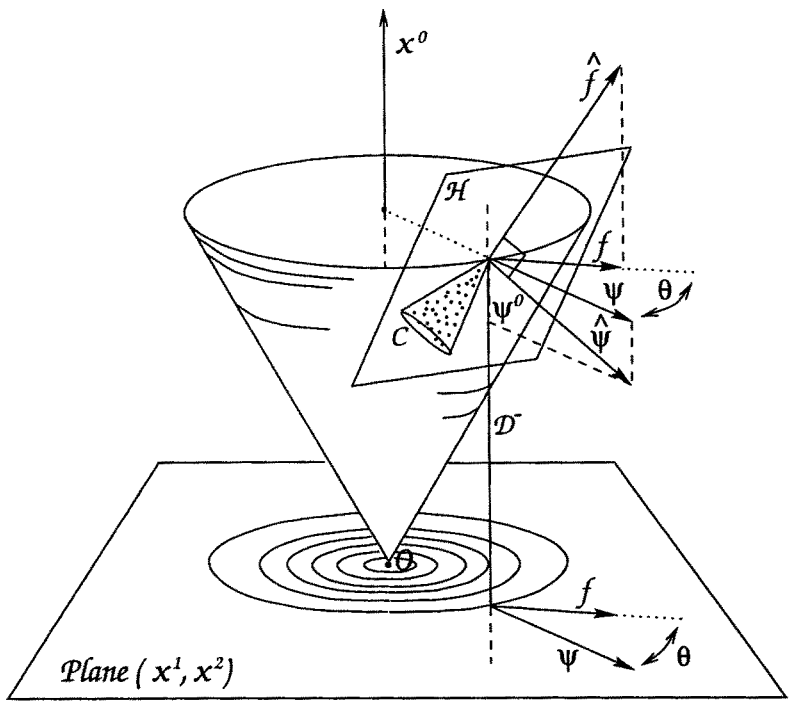


Fig. 2. Hyperplane $\mathcal{H}$ separating the half-line $\mathcal{D}^-$ and the convex cone C.

Definition 8. - *If the smooth vector mapping $\varphi = (\varphi^1, \varphi^2, \ldots, \varphi^n)$ from K to M is injective, the image $L = \varphi(K)$ is called a s-dimensional* curvilinear polyhedron *in M.*

- *Any set $\mathcal{X} \subset \Omega$ which is the union of a finite or countable number of curvilinear ploy hedra, such that only a finite number of these polyhedra intersect every closed bounded set lying in Ω, will be called a* piecewise-smooth *set in Ω. The dimension of this set will be the highest dimension of polyhedra involved in the construction.*

Remark 4. *It has been proven in [12] that any non-singular smooth surface of dimension less than n, closed in Ω, can be decomposed in a finite number of curvilinear polyhedra. Therefore, such a surface is is a piecewise-smooth set in Ω.*

Now let us state the problem: In the n-dimensional space M, we consider the following system:

$$\frac{dx^i}{dt} = f^i(x^1, \ldots, x^n, u) \quad i = 1, \ldots, n \tag{12}$$

where the control $u = (u_1, \ldots, u_m)$ belongs to an open set $U \subset \mathbf{R}^m$. The problem is the following one: *Given any two points x_0 and $x_1 \in \Omega$, among all the piecewise continuous controls $u(t)$ transferring the point from x_0 to x_1 find the one which minimizes the functional $J = \int_{t_0}^{t_1} f^0(x(t), u(t))dt$.*

Now, let us assume that are given a piecewise-smooth set N of dimension lower or equal to $n-1$, and $n+1$ piecewise-smooth sets $P^0, P^1, \ldots, P^n$ verifying

$$P^0 \subset P^1 \subset P^2 \subset \ldots \subset P^n = \Omega, \tag{13}$$

and a function v defined in Ω and taking its values in U. Now, we can introduce Boltianskii's definition of a regular synthesis.

Definition 9. *The sets, $N, P^0, \ldots, P^n$ and the function v effect a* regular synthesis *for (12) in the region Ω, if the following conditions are satisfied.*

A *The set P^0 is the target point. Every smooth component of $P^i \setminus (P^{i-1} \cup N)$, $i = 1, \ldots, n$, is an i-dimensional smooth manifold in Ω; these components will be called* i-dimensional cells. *The function v is continuously differentiable on each cell and can be extended into a continuously differentiable function on the neighbourhood of the cell.*

B *All the cells are grouped into cells of the* first *or* second *type (T_1 or T_2) in the following manner:*

(1) *If σ is a 1-dim cell of type T_1, then it is a segment of a phase trajectory solution of (12) approaching the target P^0 with a nonzero phase velocity.*

If σ is a i-dim cell of type T_1 $(i > 1)$, then through every point of σ, passes a unique trajectory solution of equation (12). Furthermore, there exists an $(i-1)$-dim cell $\Pi(\sigma)$ such that every trajectory solution of (12) leaves σ after a finite time, and strikes against the cell $\Pi(\sigma)$ at a nonzero angle and with a nonzero phase velocity.

(2) *If σ is a $(i-1)$-dim cell of type T_2 $(i \geq 1)$, then, from any point of σ there issues a unique trajectory of (12), moving in an $(i+1)$-dim cell $\Sigma(\sigma)$ of type T_1. Moreover, the function $v(x)$ is continuously differentiable on $\sigma \cup \Sigma(\sigma)$.*

(3) *All 3-dim cells are of type T_1.*

C *The conditions $B(1)$, $B(2)$ and $B(3)$ ensure the possibility of extending the trajectories solutions of (12) from cell to cell.*[2] *It is required that each trajectory "pierces" cells of the second kind only a finite number of times. In this connection every trajectory terminates at the point O. We will refer to these trajectories as being* marked.

D *From every point of the set $\Omega \setminus N$ there exists a unique marked trajectory that leads to O. From every point of N there issues a trajectory, solution of (12), not necessarily unique and which is also said to be* marked.

E *All the marked trajectories are extremals.*

F *The value of the functional J computed along the marked trajectories ending at the point O, is a continuous function of the initial point. In particular, if several trajectories start from a point x_0 of N, then, J takes the same value in each case.*

From this definition, we can now express Boltianskii's sufficient optimality condition:

Theorem 6. If a regular synthesis is effected in the set Ω under the assumption that the derivatives $\frac{\partial f^i}{\partial x^j}$ and $\frac{\partial f^i}{\partial u^k}$ exist and are continuous, and that $f^0(x,u) > 0$, then all the marked trajectories are optimal (in the region Ω).

4 Shortest paths for the Reeds-Shepp car

4.1 The pioneer works by Dubins and Reeds and Shepp

The initial work by Dubins from 1957 considered a particle moving at a constant velocity in the plane, with a constraint on the average curvature of trajectories. Using techniques close to those involved in the proof of Fillipov's existence theorem, Dubins proved the existence of shortest paths for his problem. He showed that the optimal trajectories are necessarily made up with arc of circles

[2] Trajectories are extended from the cell σ into $\Pi(\sigma)$ if $\Pi(\sigma)$ is of type T_1, and from σ to $\Sigma(\Pi(\sigma))$ if $\Pi(\sigma)$ is of type T_2.

C of minimal turning radius and line segments S. Therefore, he proved that any optimal path must be of one of the following two path types:

$$\{C_aC_bC_e\ , C_aS_dC_e\} \quad \text{where: } 0 \leq a, e < 2\pi,\ \pi < b < 2\pi, \text{ and } d \geq 0 \qquad (14)$$

In order to specify the direction of rotation the letter C will sometimes be replaced by a "r" (right turn) or a "l" (left turn). The subscripts a, e, ... specify the length of each elementary piece.

Later, Reeds and Shepp [31] considered the same problem, when backwards motions are allowed ($|u_1| = k$). In both cases, as the modulus of the linear velocity keeps constant, the shortest path problem is equivalent to the minimum-time problem. Contrary to Dubins, Reeds and Shepp did not prove the existence of optimal paths. Indeed, as the control set is no more convex the existence of optimal paths cannot be deduced directly from Fillipov's theorem. From Dubins's result, they deduced that any subpath of an optimal path, lying between two consecutive cusp-points, must belong to the sufficient family (14). Finally, they proved that the search for a shortest path may be restricted to the following sufficient family (the symbol | indicates a cusp):

$$\begin{array}{l}\{\ C|C|C,\ \ CC|C,\ \ C|CC,\ \ CC_a|C_aC,\ \ C|C_aC_a|C,\\ C|C_{\pi/2}SC,\ \ CSC_{\pi/2}|C,\ \ C|C_{\pi/2}SC_{\pi/2}|C,\ \ CSC\ \}\end{array} \qquad (15)$$

However the techniques used by Reeds and Shepp in their proof are based on specific *ad hoc* arguments from differential calculus and geometry, specially developed for this study, and cannot constitute a framework for further studies.

The following two subsections present a sequence of more recent works based on optimal control theory and geometry which have led to characterize the complete solution of Reeds and Shepp's problem. Section (4.2) presents a result simultaneously obtained by Sussmann and Tang [36] on the one hand, and by Boissonnat, Cerezo and Leblond [2] on the other hand, showing how Reeds and Shepp's result can be found and even slightly improved by using optimal control theory.

Section (4.3) presents a work by Souères and Laumond [33] who achieved the characterization of shortest paths by completing the local reasoning of PMP with global geometric arguments.

The last section (section 4.4) concludes the study by providing, a posteriori, a new proof of the construction by using Boltianskii's sufficient conditions [35].

4.2 Characterization of a sufficient family: A local approach

This section summarizes the work by Sussmann and Tang [36]; we use the notations introduced by the authors.

The structure of commutations As the control set $U_{RS} = \{-1, +1\} \times [-1, 1]$ related to Reeds and Shepp's problem (RS) is not convex, it is not possible to deduce the existence of optimal paths directly from Fillipov's existence theorem. For this reason, the authors have chosen to consider what they call the *convexified problem* (CRS) corresponding to the convexified control set $U_{CRS} = [-1, +1] \times [-1, +1]$ for which Fillipov's existence theorem (theorem 2) applies. The existence of optimal paths for RS will be established *a posteriori* as a byproduct.

Let $g_1(x) = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix}$, and $g_2(x) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ denote the two vector fields on which the kinematics of the point is defined. With this notation system (2) becomes :

$$\dot{x} = f(x, u) = g_1(x)\, u_1 + g_2(x)\, u_2 \tag{16}$$

the corresponding hamiltonian is:

$$H = < \psi, f > = \psi_1 \cos\theta\, u_1 + \psi_2 \sin\theta\, u_1 + \psi_3\, u_2 = \phi_1\, u_1 + \phi_2\, u_2$$

where $\phi_1 = < \psi, g_1 >$, and $\phi_2 = < \psi, g_2 >$ represent the switching functions. From PMP, a necessary condition for $(u_1(t), u_2(t))$ to be an optimal control law, is that there exists a constant $\psi_0 \leq 0$ such that at each time $t \in [t_0, t_1]$

$$\begin{aligned} -\psi_0 &= < \psi(t), g_1(x(t)) > u_1(t) + < \psi(t), g_2(x(t)) > u_2(t) \\ &= \max_{v=(v_1,v_2)\in U}(< \psi(t), g_1(x(t)) > v_1(t) + < \psi(t), g_2(x(t)) > v_2(t)) \end{aligned} \tag{17}$$

$$\text{and } \dot{\psi}(t) = \frac{-\partial H}{\partial x}(\psi(t), x(t), u(t)) = -\psi(t)\, [u_1 \frac{dg_1}{dx} + u_2 \frac{dg_2}{dx}]$$

A necessary condition for t to be a switching time for u_i is that $\phi_i = 0$. Therefore, on any subinterval of $[t_0, t_1]$ where the switching function ϕ_i does not vanish the corresponding control component u_i is *bang* i.e. maximal or minimal.

Now, by means of theorem 4 we can express the derivative of the switching functions in terms of Lie brackets:

$$\begin{aligned} \phi_1 =< \psi, g_1 > &\Longrightarrow \dot{\phi}_1 = u_2 < \psi, [g_2, g_1] > \\ \phi_2 =< \psi, g_2 > &\Longrightarrow \dot{\phi}_2 = -u_1 < \psi, [g_2, g_1] > \end{aligned}$$

Thus the function $\phi_3 = <\psi, g_3>$, where $g_3 = [g_1, g_2] = (-\sin\theta, \cos\theta, 0)^T$, seems to play an important role in the search for switching times. We have then:

$$\dot{\phi}_1 = u_2\phi_3 \ , \ \dot{\phi}_2 = -u_1\phi_3 \ , \ \dot{\phi}_3 = -u_2\phi_1 \tag{18}$$

Maximizing the Hamiltonian leads to:

$$u_1(t) = \text{sign}(\phi_1(t)) \quad \text{and} \quad u_2(t) = \text{sign}(\phi_2(t)) \tag{19}$$

$$\text{where} \quad \text{sign}(s) = \begin{cases} +1 & \text{if } s > 0 \\ -1 & \text{if } s < 0 \\ \text{any element of [-1,1]} & \text{if } s = 0 \end{cases}$$

Then from PMP we get:

$$|\phi_1(t)| + |\phi_2(t)| + \psi_0 = 0 \tag{20}$$

On the other hand, at each point of the manifold M, the vector fields g_1, g_2 and g_3 define a basis of the tangent space. Therefore, as the adjoint vector never vanishes, ϕ_1, ϕ_2 and ϕ_3 cannot have a common zero. It follows that:

$$|\phi_1(t)| + |\phi_2(t)| + |\phi_3(t)| \neq 0 \tag{21}$$

Equations (18), (19), (20) and (21) define the structure of commutations; several properties may be deduced from these relations as follows:

Lemma 1. There do not exist (non trivial) abnormal extremals for CRS.

Proof: If $\psi_0 = 0$ then (20) $\Longrightarrow \phi_1 \equiv 0$ and $\phi_2 \equiv 0$. Then (21) $\Longrightarrow \phi_3 \neq 0$ but (18) $\Longrightarrow u_1\phi_3 = u_2\phi_3 = 0 \Longrightarrow u_1 = u_2 = 0$. The only remaining possibility is a trivial trajectory i.e. of zero length in zero time □.

Lemma 2. On a non trivial extremal trajectory for CRS, ϕ_1 and ϕ_2 cannot have a common zero.

Proof: If $\exists t \in [t_0, t_1]$ such that $\phi_1(t) = \phi_2(t) = 0$ then (20) $\Longrightarrow \psi_0 = 0$, we conclude with lemma 1 □

Lemma 3. Along an extremal for CRS $\kappa = \phi_1^2 + \phi_3^2$ is constant. Furthermore, $(\kappa = 0) \Longleftrightarrow (\phi_1 \equiv 0)$

Proof: As $\dot{\phi_1} = u_2\phi_3$ and $\dot{\phi_3} = u_2\phi_1$ we deduce that $\kappa = \phi_1^2 + \phi_3^2$ is constant. $\kappa = 0 \Rightarrow \phi_1 \equiv 0$, obvious. Inversely, suppose $\phi_1 \equiv 0$ but $\kappa \neq 0$; from lemma 2 it follows that $\phi_2 \neq 0$. Therefore as $u_2 = sign(\phi_2)$, $u_2 \neq 0$. But then (18) $\Longrightarrow 0 = \dot{\phi_1} = u_2.\phi_3 \Longrightarrow \phi_3 = 0$ that leads to a contradiction □.

Lemma 4. Along an extremal for CRS, either the zeros of ϕ_1 are all isolated, or $\phi_1 \equiv 0$. Furthermore, at an isolated zero of ϕ_1, $\dot{\phi_1}$ exists and does not vanish.

Proof: Let $x(t,u)$ be an extremal for CRS on $[t_0,t_1]$. Suppose $\phi_1 \not\equiv 0$; it follows that $\kappa > 0$. Let $\tau \in]t_0,t_1[$ such that $\phi_1(\tau) = 0$, from lemma 2 $\phi_2(\tau) \neq 0$ and as ϕ_2 is continuous there exists a subinterval $I \subset]t_0,t_1[$ containing τ such that $\forall t \in I, \phi_2(t) \neq 0$. Therefore the sign of ϕ_2 is constant on I. From this, we deduce that $u_2 \equiv 1$ or $u_2 \equiv -1$ on I. In either case, since the equation $\dot{\phi_1} = u_2(t)\ \phi_3(t)$ holds on I, it comes $\dot{\phi_1}(t) = \pm\phi_3(t)$, the sign keeping constant on I. $\kappa > 0$ and $\phi_1(\tau) = 0 \Longrightarrow \phi_3(\tau) \neq 0 \Longrightarrow \dot{\phi_1}(\tau) = \pm\phi_3(\tau) \neq 0$. therefore, τ is an isolated zero. □

Therefore, there exist two kinds of extremal trajectories for CRS:

- type A: trajectories with a finite number of switch on u_1,
- type B: trajectories along which $\phi_1 \equiv 0$ and either $u_2 \equiv 1$ or $u_2 \equiv -1$.

Before starting the study of these two path types, we need to state a simple preliminary lemma.

Lemma 5. If an optimal trajectory for CRS is an arc bang C_a then necessarily $a \leq \pi$.

Proof: When $a > \pi$ it suffice to follow the arc of length $2\pi - a$ in the opposite direction. □

Characterization of type A trajectories First, let us consider the type A trajectories with **no cusp** i.e. trajectories along which $u_1 \equiv 1$ or $u_1 \equiv -1$. According to the symmetry of the problem we can restrict the study to the case that $u_1 \equiv 1$. The corresponding trajectories are the solutions of Dubins' problem (DU) which are optimal for CRS.

Let $\gamma(t), t \in [t_0,t_1]$ be such a trajectory. From lemma 4 we know that the zeros of ϕ_1 are all isolated. Furthermore, ϕ_1 cannot vanish on $]t_0,t_1[$ because in that case the sign of ϕ_1 would have to change, and u_1 would have to switch. Therefore, as γ is a trajectory for DU, $\phi_1(t) \geq 0$ on $[t_0,t_1]$ and $\phi_1 > 0$ on $]t_0,t_1[$. Equations (18) become: $\dot{\phi_2} = -\phi_3$ and $\dot{\phi_3} = -u_2\phi_1$. Then ϕ_2 is of class C^1 and $\ddot{\phi_2} = u_2\phi_1$. Furthermore, $u_2 =$ sign(ϕ_2), then $\ddot{\phi_2} = \phi_1$sign(ϕ_2). This means that ϕ_2 is convex, (resp. concave) on the whole interval where $\phi_2 > 0$ (resp. $\phi_2 < 0$). From that we deduce the following property:

Lemma 6. A trajectory γ with no cusp, optimal for CRS, is necessarily of one of the following three kinds:

- $C_a \quad 0 \leq a \leq \pi$
- $C_a C_b \quad 0 < a \leq \frac{\pi}{2}$ and $0 < b \leq \frac{\pi}{2}$
- $C_a S_c C_b \quad 0 < c\,, \quad 0 \leq a \leq \frac{\pi}{2}$ and $0 \leq b \leq \frac{\pi}{2}$

Proof:

As before we consider only the trajectories along which $u_1 \equiv 1$. If ϕ_2 does not vanish, γ is an arc bang; from lemma 5 we can conclude.

If ϕ_2 vanishes, let us denote by I the time interval defined by $I = \{t, t \in [t_0, t_1], \phi_2(t) \neq 0\}$. As $\phi_2^{-1}(0)$ is closed, I is relatively open in $[t_0, t_1]$. Let $\mathcal{I}$ be the set of connected components of I. First, let us prove that $\mathcal{I}$ does not contain any open interval $J =]t', t''[\subset [t_0, t_1]$. Indeed, if J is such an interval, then $\phi_2(t') = \phi_2(t'') = 0$. Since ϕ_2 is either negative and concave, or positive and convex on J, then necessarily $\phi_2 \equiv 0$ on J which is a contradiction. So $[t_0, t_1]$ is partitioned into at most three intervals I_1, I_2, I_3 such that ϕ_2 never vanishes on $I_1 \cup I_3$, and $\phi_2 \equiv 0$ on I_2. On each interval J in $\mathcal{I}$ u_2 is constant and equals 1 or -1. From equations (18) we get $\ddot{\phi}_3 + \phi_3 = 0$. We have shown that ϕ_2 vanishes on $I_2 = [t', t'']$; if $t_0 < t'$ then ϕ_2 is convex and positive (or concave and negative) on $[t_0, t'[$ and $\phi_2(t') = 0$. Therefore both ϕ_3 and $\dot{\phi}_3$ have a constant sign on $]t_0, t'[$ (for instance, if $\phi_2 > 0$ on $[t_0, t'[$, then $\dot{\phi}_3 = -u_2\phi_1$ and $u_2 = -sign(\phi_2) = -1 \Rightarrow \dot{\phi}_3 = \phi_1 \neq 0$). Also $\phi_3 = -\dot{\phi}_2$, so ϕ_3 has no zero either because the derivative of a convex function only vanishes at its minimum. This implies that $t' - t_0 \leq \frac{\pi}{2}$. By applying a same reasoning on $]t'', t_1]$ we conclude the proof for the case $u_1 \equiv 1$. The case $u_1 \equiv -1$ can be derived from the same arguments □.

Remark 5. *The previous lemma does not solve Dubins' problem. It just determines the structure of Dubins' trajectories which are CRS-optimal. Indeed, a time optimal trajectory for DU is not necessarily optimal for CRS.*

Now let us go back to the general form of type A trajectories. By integrating the adjoint equations type A trajectories may be very well characterized. Let us consider the adjoint system:

$$\begin{cases} \dot{\psi}_1 = -\frac{\partial H}{\partial x} = 0 \\ \dot{\psi}_2 = -\frac{\partial H}{\partial y} = 0 \\ \dot{\psi}_3 = -\frac{\partial H}{\partial \theta} = \psi_1 \sin\theta\, u_1 - \psi_2 \cos\theta\, u_1 = \psi_1\, \dot{y} - \psi_2\, \dot{x} \end{cases}$$

Hopefully, these equations are easily integrable: ψ_1 and ψ_2 are constant on $[t_0, t_1]$ and if we suppose $x(t_0) = y(t_0) = 0$ we get $\phi_2(t) = \psi_3(t) = \psi_3(t_0) + \psi_1 y(t) - \psi_2 x(t)$. Therefore, from relation (17) we can deduce that the switching points are located on three parallel lines.

- when $\phi_2(t) = 0$, the point lies on the line D_0: $\psi_1 y - \psi_2 x + \psi_3(t_0) = 0$
- when $\phi_1(t) = 0$, we deduce from (17) that $\psi_3(t)u_2(t) + \psi_0 = 0$:
  - If $u_2 = 1$ the point is on the line $D^+ : \psi_1 y - \psi_2 x + \psi_3(t_0) + \psi_0 = 0$
  - If $u_2 = -1$ the point is on the line $D^- : \psi_1 y - \psi_2 x + \psi_3(t_0) - \psi_0 = 0$

- If $\phi_2(t)$ vanishes over a non empty interval $[\tau_1, \tau_2] \subset [t_0, t_1]$, we get from relation (17): $\psi_1 \cos\theta(t) + \psi_2 \sin\theta(t) + \psi_0 = 0$. According to lemma 2, the constant ψ_1 and ψ_2 cannot be both zero, it follows that θ remains necessarily constant on $[\tau_1, \tau_2]$, the singular control component u_2 is equal to 0, and the corresponding trajectory is a segment of D_0.

- At a cusp point $\phi_1 = \psi_1 \cos\theta + \psi_2 \sin\theta = 0$. It follows that the point is oriented perpendicularly to the common direction of the three lines.

To sum up, type A trajectories are made up with arcs of circle (C) of minimal turning radius which correspond to the regular control laws ($u_1 = \pm 1, u_2 = \pm 1$) and line segments (S) which correspond to the singularity of the second control component: ($u_1 = \pm 1, u_2 = 0$). The line segments and the point of inflection are on D_0, the cusp point are on D^+ or D^- and at each cusp the direction of the point is perpendicular to the common direction of the lines, see figure (3). We have the following lemma:

Lemma 7. In the plane of the car's motion any trajectory of type A is located between two parallel lines D^+ and D^-. The points of inflection and the line segments belong to a third line D_0 having the same direction as D^+ and D^- and located between them at equal distance $|\frac{\psi_0}{\psi_1}| \leq \frac{\pi}{2}$. The cusp-points are located on D^+ when $u_2 = 1$ and on D^- when $u_2 = -1$; at a cusp point the representative point's orientation is perpendicular to these lines.

At this stage, by using a geometric reasoning it is possible to prove that some sequences of arcs and line segments are never optimal.

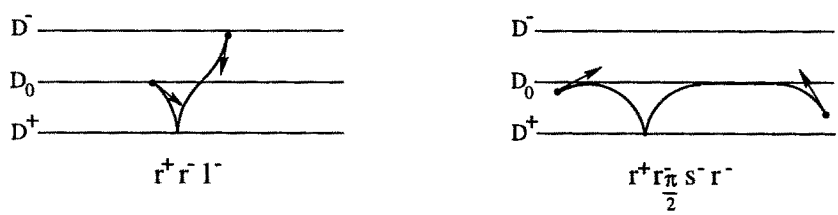


Fig. 3. Optimal paths of type (**A**) lie between two parallel lines D^+ and D^-.

Lemma 8. The following trajectories cannot be CRS-optimal paths.

1. $C_a|C_\pi \quad a > 0$
2. $C_a|C_{\frac{\pi}{2}}S_eC_{\frac{\pi}{2}} \quad a > 0, e \geq 0$, with the same direction of rotation (l or r) on the arcs $C_{\frac{\pi}{2}}$ located at each extremity of S.
3. $C_{\frac{\pi}{2}}S_eC_{\frac{\pi}{2}}|C_{\frac{\pi}{2}}$, $e \geq 0$, with an opposite direction of rotation on the arcs $C_{\frac{\pi}{2}}$ located at each extremity of S.
4. $C_a|C_bC_b|C_bC_b \quad a, b > 0$

Proof: The method of the proof consists in showing that each trajectory is equivalent to another one which is clearly not optimal. The reasoning is illustrated at figure (4): By replacing a part of the initial path by the equivalent one drawn in dotted line, we get an equivalent trajectory i.e. having the same cost and linking the same two configurations. Then, using the preceding lemmas 6 and 7 we prove that this new trajectory does not verify the necessary conditions of PMP. On figure (4) the path 1, $l_a^+l_\pi^-$ is equivalent to a path $l_a^+r_\pi^+$ and from lemma 6 we now that such a path is not optimal. The three other path types (2, 3 and 4) do not satisfy lemma 7. Indeed, either the points of inflection and the line segment do not belong to the same line D_0 or the direction of the point at a cusp is not perpendicular to D_0. □

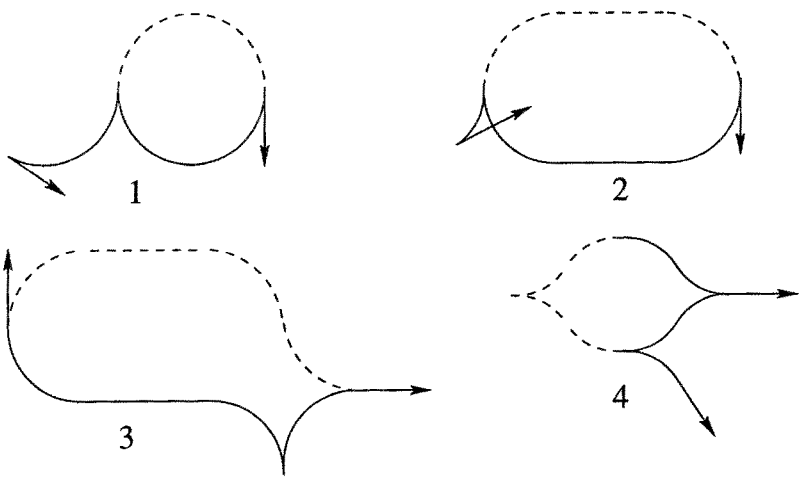


Fig. 4. Non-optimal equivalent trajectories

Finally, Using the theory of envelopes, Sussmann and Tang showed that a path $C_a|C_bC_b|C_b$ is never optimal. Due to the lack of space we cannot present here this technical part of the proof, the reader will have to refer to [36]. This last result eliminates type A trajectories with more than two cusps. The remaining possible sequences of (C) and (S) determine eight path types which

are represented by the types (II) to (IX) of the sufficient family (22) presented at section 4.2.

Characterization of type B trajectories Let us first consider the case that $u_2 \equiv 1$; we call this subproblem LTV (Left Turn Velocity). In order to lead their reasoning the authors considered what they called the lifted problem LLTV (Lifted Left Turn Velocity) obtained from LTV by regarding the variable θ as a real number x_3. For this last problem, as $\dot{x_3} \equiv 1$, any trajectory γ linking the point x_0 at time t_0 to the point x_1 at time t_1 has the cost: $\Delta t = x_3(t_1) - x_3(t_0)$. The same phenomenon occurs for LTV, but only for trajectories whose duration is lower or equal to π. For this reason the problem LLTV is called degenerate whereas the problem LTV is locally degenerate.

Using the techniques presented in chapter 1 it is straight forward to deduce from the structure of the Lie algebra $L = Lie(g_1, g_2)$ generated by the vector fields g_1 and g_2, that the problem LTV has the accessibility property. Nevertheless, as the corresponding system is not symmetric (i.e. an admissible trajectory followed backwards is not necessary admissible) we cannot deduce directly the controllability of LTV. This can be done by considering the "bang-bang" system (BB) corresponding to the control set $(u_1, u_2) \in \{-1, +1\} \times \{-1, +1\}$. As the BB system has the accessibility property and is symmetric on the connected manifold $\mathbf{R}^2 \times S^1$ it is controllable. Any admissible trajectory for BB is a sequences of tangent arcs C. By replacing every arc r_α by the complementary part $l_{2\pi-\alpha}$ followed backwards, we can transform any BB trajectory into an admissible trajectory for LTV. Therefore, we deduce the controllability of the problem LTV. This is no more true for the problem LLTV in $\mathbf{R}^3$. It suffice to note that no point $(x, y, 0)$ verifying $(x, y) \neq (0, 0)$ is reachable from the origin.

By using the Ascoli-Arzelà theorem, Sussmann and Tang proved that the reachable set for LLTV from x_0, $\mathcal{R}(x_0)$, is a closed subset of $\mathbf{R}^3$. Now, let L_0 be the ideal of the lie algebra L generated by g_1. As L_0 is the smallest linear subspace S of L, such that $\forall X \in S, \forall Y \in L, [X, Y] \in S$. It follows that $L_0 = Lie(g_1, g_3)$.

Definition 10. *(Sussmann) - L_0 is called* strong accessibility Lie algebra.

- Let $x \in \mathbf{R}^3$, $L_0(x) = span(g_1(x), g_2(x))$; a trajectory of LLTV is called a strong extremal *if the corresponding adjoint vector ψ is not trivial on $L_0(\gamma(t))$, i.e. the projection of $\psi(t)$ on $L_0(\gamma(t))$ never vanishes.*

- We will call boundary trajectory *of LLTV any trajectory $\gamma : [t_0, t_1] \to \mathbf{R}^3$ such that $\gamma(t_1)$ belongs to the boundary $\partial\mathcal{R}(x_0)$ of $\mathcal{R}(x_0)$.*

Lemma 9. Any boundary trajectory of LLTV is a strong extremal of the form $l_a^{s_0} l_\pi^{s_1} \dots l_\pi^{s_k} l_b^{s_{k+1}}$ where $0 \leq a, b \leq \pi$ and the signs $s_i \in \{+, -\}$ alternate.

Proof: Let $\gamma : [t_0, t_1] \rightarrow \mathbf{R}^3$ be a boundary trajectory for LLTV, $x_0 = \gamma(t_0)$ and $x_1 = \gamma(t_1) \in \partial\mathcal{R}(x_0)$. From theorem 5 we know that there exists a nontrivial adjoint vector $\psi = (\psi_0, \psi_1, \ldots, \psi_n)$ verifying $\psi_0 \equiv 0$. From relation (20) we get $|\phi_1| + |\phi_2| = 0$. On the other hand, $\kappa = \phi_1(t)^2 + \phi_3(t)^2 \neq 0$ otherwise $\phi_1 = \phi_2 = \phi_3 = 0$. It follows that $< \psi, g_1 >= \phi_1$ and $< \psi, g_2 >= \phi_3$ do not vanish simultaneously and therefore γ is a strong extremal for LLTV. Now, as $u_2 \equiv 1$ we know from equations (18) that ϕ_1 must be a nontrivial solution of $\ddot{\phi_1} + \phi_1 = 0$. Therefore, the distance between two consecutive zeros of ϕ_1 is exactly π and the sign of ϕ_1 changes at each switch. □

Lemma 10. Given any path γ solution of LLTV, there exists an equivalent solution γ' of LLTV which is a concatenation of a boundary trajectory and an arc bang of LLTV for the control $u_1 \equiv 1$.

Proof: Let γ be defined on $[t_0, t_1]$, $\gamma(t_0) = x_0$ and $\gamma(t_1) = x_1$ a trajectory solution of LLTV. If $x_1 \in \partial\mathcal{R}(x_0)$ the conclusion follows. Suppose now that $x_1 \in Int(\mathcal{R}(x_0))$. let us consider an arc bang for LLTV corresponding to the control $u_1 \equiv 1$, ending at x_1. As the set $\mathcal{R}(x_0)$ is closed, by following this path backwards from x_1 the representative point reaches, after a finite time, a point x_1' belonging to the boundary. The problem being degenerate, the trajectory γ made up by the boundary trajectory from x_0 to x_1' followed by the arc bang from x_1' to x_1 is equivalent to γ. □

Now Suppose $\gamma : [t_0, t_1] \rightarrow \mathbf{R}^2 \times S^1$ is an LTV trajectory time optimal for CRS. Let $\pi : \mathbf{R}^3 \rightarrow \mathbf{R}^2 \times S^1$ be the canonical projection, then $\gamma = \pi \circ \gamma^\star$ where $\gamma^\star : [t_0, t_1] \rightarrow \mathbf{R}^3$ is a trajectory of LLTV. From the previous lemma we can replace $\gamma^\star$ by the concatenation $\gamma^\star_{new}$ of a boundary trajectory $\gamma_1^\star$ and a bang trajectory $\gamma_2^\star$ for $u_1 \equiv 1$, and then project these down to trajectories γ_{new}, γ_1, γ_2 in $\mathbf{R}^2 \times S^1$. Then γ_1 is of the form $l_a^{s_0} l_\pi^{s_1} \ldots l_\pi^{s_k} l_b^{s_{k+1}}$. But from lemma 8 a path $C_a|C_\pi$ with $a > 0$ cannot be optimal. It follows that γ_1 contains at most one cusp, and the length of γ_2 is lower or equal to π. Therefore, γ_{new} is a path $l_a^+ l_b^- l_c^+$ or $l_a^- l_d^+ l_e^+ = l_a^- l_{d+e}^+$ with a, b, c and $d + e$ at most equal to π. Hence, the type $l_a^+ l_b^- l_c^+, 0 \leq a, b, c \leq \pi$ constitutes a sufficient family for LLTV.

Using the same reasoning for the dual problem RTV (Right Turn Velocity), the path type $r_a^+ r_b^- r_c^+$ with $0 \leq a, b, c \leq \pi$ appears to be also sufficient.

Remark 6. *The reasoning above cannot be directly held in* $\mathbf{R}^2 \times S^1$ *for LTV because in this case the length of a trajectory steering the point from* x_0 *to* x_1 *is not necessarily equal to* $\theta(t_1) - \theta(t_0)$.

Sufficient family for RS From the reasoning above it appears that the search for optimal trajectories for CRS may be restricted to the following sufficient family containing nine path types:

$$
\begin{array}{lll}
(I) & l_a^+ l_b^- l_e^+ \text{ or } r_a^+ r_b^- r_e^+ & 0 \le a \le \pi\,,\ 0 \le b \le \pi\,,\ 0 \le e \le \pi \\
(II)(III) & C_a|C_bC_e \text{ or } C_aC_b|C_e & 0 \le a \le b\,,\ 0 \le e \le b\,,\ 0 \le b \le \frac{\pi}{2} \\
(IV) & C_aC_b|C_bC_e & 0 \le a < b\,,\ 0 \le e < b\,,\ 0 \le b \le \frac{\pi}{2} \\
(V) & C_a|C_bC_b|C_e & 0 \le a < b\,,\ 0 \le e < b\,,\ 0 \le b \le \frac{\pi}{2} \\
(VI) & C_a|C_{\frac{\pi}{2}}S_lC_{\frac{\pi}{2}}|C_b & 0 \le a < \frac{\pi}{2}\,,\ 0 \le b < \frac{\pi}{2}\,,\ 0 \le l \\
(VII)(VIII) & C_a|C_{\frac{\pi}{2}}S_lC_b \text{ or } C_bS_lC_{\frac{\pi}{2}}|C_a & 0 \le a \le \pi\,,\ 0 \le b \le \frac{\pi}{2}\,,\ 0 \le l \\
(IX) & C_aS_lC_b & 0 \le a \le \frac{\pi}{2}\,,\ 0 \le l\,,\ 0 \le b \le \frac{\pi}{2}
\end{array}
\tag{22}
$$

However, all the path contained in this family are obtained for $u_1 = 1$ or $u_1 = -1$, and by this, are admissible for RS. Therefore, this family constitutes also a sufficient family for RS which contains 46 path types. This result improves slightly the preceding statement by Reeds and Shepp of a sufficient family containing 48 path types.

On the other hand, as Fillipov's existence theorem guarantees the existence of optimal trajectories for the convexified problem CRS, it ensures the existence of shortest paths with bounded curvature radius for linking any two configurations of Reeds and Shepp's car. Applying PMP to Reeds and Shepp's problem we deduce the following lemma that will be useful in the sequel.

Lemma 11. (Necessary conditions of PMP)
Optimal trajectories for RS are of two types:

- **A/** Paths lying between two parallel lines D^+ and D^- such that the straight line segments and the points of inflection lie on the median line D_0 of both lines, and the cusp points lie on D^+ or D^-. At a cusp the point's orientation is perpendicular to the common direction of the lines (see figure 3),
- **B/** Paths $C|C|\ldots|C$ with $length(C) \le \pi$ for any C.

4.3 A geometric approach: construction of a synthesis of optimal paths

Symmetry and reduction properties In order to analyse the variation of the car's orientation along the trajectories let us consider the variable θ as a real number. To a point $q = (x, y, \theta^*)$ in $\mathbf{R}^2 \times S^1$ correspond a set $Q = \{(x, y, \theta) \ / \ \theta \in \theta^*\}$ in $\mathbf{R}^3$ where θ^* is the class of congruence modulus 2π. Therefore, the search for a shortest path from q to the origin in $\mathbf{R}^2 \times S^1$ is equivalent to the search for a shortest path from Q to the origin in $\mathbf{R}^3$. By considering the problem in $\mathbf{R}^3$ instead of $\mathbf{R}^2 \times S^1$ we can point out some interesting symmetry properties. First let us consider trajectories starting from each horizontal plane $P_\theta = \{(x, y, \theta),\ x, y \in \mathbf{R}^2\} \subset \mathbf{R}^3$.

In the plane P of the robot's motion, or in the plane P_θ, we denote by Δ_θ the line of equation: $y = -x \cot \frac{\theta}{2}$ and $\Delta_\theta^\perp$ the line perpendicular to Δ_θ passing through 0. Given a point (M, θ), we denote by M_1 the point symmetric to M with respect to O, M_2 the point symmetric to M with respect to Δ_θ, and M_3 the point symmetric of M_1 with respect to Δ_θ. Let $\mathcal{T}$ a be path from (M, θ) to $(O, 0)$.

Lemma 12. There exist three paths $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ each isometric to $\mathcal{T}$, starting respectively from (M_1, θ), (M_2, θ) and (M_3, θ) and ending at $(O, 0)$ (see figure 5).

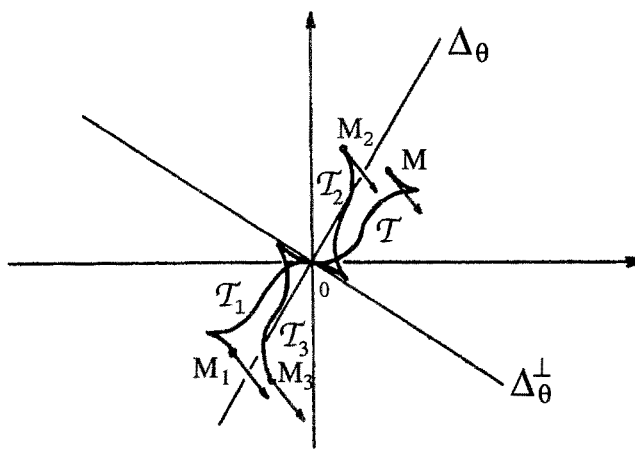


Fig. 5. A path gives rise to 3 isometric ones.

Proof: (see Figure 5) $\mathcal{T}_1$ is obtained from $\mathcal{T}$ by the symmetry with respect to O.

Proving the existence of $\mathcal{T}_2$ requires us to consider the construction illustrated at figure (6): We denote by δ the line passing through M and making an angle θ with the x-axis, and s the axial symmetry with respect to δ. Let A be the intersecting point of δ with the x-axis and r the rotation by the angle $-\theta$ around A. Let us note $L = r(M)$. Finally, t, represents the translation of vector $\overrightarrow{LO}$. We denote by $\mathcal{T}_2$ the image of $\mathcal{T}$ by the isometry $\Im = t \circ r \circ s$. $\mathcal{T}_2$ links the directed point $(\tilde{M}, \tilde{\theta}) = \Im((O, 0))$ to $(O, 0) = \Im(M, \theta)$. $\tilde{\theta}$ clearly equals θ. We have to prove that $\tilde{M} = M_2$. Let respectively α and β be the angles made by (O, M) and $(O, \tilde{M})$ with the x-axis. The measure of the angle made by the bisector of $\widehat{(M, O, \tilde{M})}$ and the x-axis is: $\alpha + \frac{\beta - \alpha}{2} = \frac{\alpha + \beta}{2} = \frac{\pi - \theta}{2}$. As $\tan \frac{\pi - \theta}{2} = -\cot \frac{\theta}{2}$, we can assert that $\tilde{M}$ is the symmetric point of M with respect to Δ_θ, i.e. M_2.

Finally $\mathcal{T}_3$ is obtained as the image of $\mathcal{T}$ by $\Im$ followed by the symmetry with respect to the origin. □

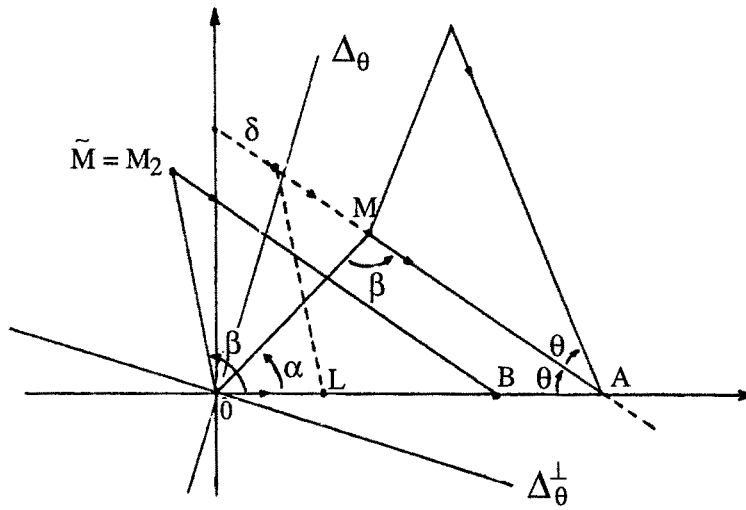


Fig. 6. Construction of the isometry $\Im$.

Lemma 13. If $\mathcal{T}$ is a path from $(M(x,y),\theta)$ to $(O,0)$, there exists a path $\overline{\mathcal{T}}$, isometric to $\mathcal{T}$, from $(\overline{M}(x,-y),-\theta)$ to $(O,0)$.

Proof: It suffices to consider the symmetry s_x with respect to the abscissa axis.

Remark 7. – *By combining the symmetry with respect to Δ_θ and the symmetry with respect to O, the line $\Delta_\theta^\perp$ appears to be also an axis of symmetry. According to lemmas 12 and 13 it is enough to consider paths starting from one quadrant in each plane P_θ, and only for positive or negative values of θ.*

– *The constructions above allow us to deduce easily the words w_1, w_2, w_3 and w_4 describing $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ and $\mathcal{T}_4$ from the word w describing $\mathcal{T}$.*

- *w_1 is obtained by writing w, then by permutating the superscripts $+$ and $-$*
- *w_2 is obtained by writing w in the reverse direction, then by permutating the superscripts $+$ and $-$*
- *w_3 is obtained by writing w in the reverse direction*
- *$\overline{w}$ is obtained by writing w, then by permutating the r and the l* □

As a consequence of both lemmas above a last symmetry property holds in the case that $\theta = \pm\pi$:

Lemma 14. If $\mathcal{T}$ is a path from $(M(x,y),\pi)$ (resp. $(M(x,y),-\pi)$) to $(O,0)$, there exists an isometric path $\mathcal{T}'$ from $(M(x,y),-\pi)$ (resp. $(M(x,y),\pi)$) to $(O,0)$.

The word w' describing $\mathcal{T}'$ is obtained by writing w in the opposite direction, then by permutating on the one hand the r and the l, and on the other hand the $+$ and $-$.

Remark 8. *The points $(M(x,y),\pi)$ and $(M(x,y),-\pi)$ represent the same configuration in $\mathbf{R}^2 \times S^1$ but are different in $\mathbf{R}^3$. This means that the trajectories $\mathcal{T}$ and $\mathcal{T}'$ are isometric and have the same initial and final points, but along these trajectories the car's orientation varies with opposite direction.*

Proof of lemma 14: We use the notation of lemma 12 and 13. Let $(M(x,y),\pi)$ be a directed point and $\mathcal{T}$ a trajectory from (M,π) to $(0,0)$. When $\theta = \pm\pi$ the axis Δ_θ is aligned with the x-axis. By lemma 12, there exists a trajectory $\mathcal{T}_2 = \Im(\mathcal{T})$, isometric to $\mathcal{T}$, starting at $(M_2(x,-y),\pi)$ and ending at $(O,0)$. Then by lemma 12 there exists a trajectory $\overline{\mathcal{T}_2} = s_x(\mathcal{T}_2)$, isometric to $\mathcal{T}_2$, starting at $(\overline{M_2}(x,y),-\pi)$ and ending at $(O,0)$. Let us call $\mathcal{T}'$ the trajectory $\overline{\mathcal{T}_2}$, then $\mathcal{T}' = s_x \circ \Im(\mathcal{T})$ is isometric to $\mathcal{T}$ and by combining the rules defining the words w_2 and $\overline{w}$ we obtain the rule characterizing $\overline{w_2} = w'$ (the same reasoning holds when $\theta = -\pi$.) □

Now by using lemma 14 we are going to prove that it suffice to consider paths starting from points (x,y,θ) when $\theta \in [-\pi,\pi]$. In the family (22) three types of path may start with an initial orientation θ that does not belong to $[-\pi,\pi]$. These types are (I) and (VII) & (VIII). Combining lemma 14 with the necessary condition given by PMP we are going to refine the sufficient family (22) by rejecting those paths along which the total angular variation is greater than π.

Lemma 15. In the family (22), types *(I), (VII)* and *(VIII)* may be refined as follows:

$$\text{(I)} \qquad l_a^+ l_b^- l_e^+ \text{ or } r_a^+ r_b^- r_e^+ \qquad 0 \le a+b+e \le \pi$$

$$\text{(VII)(VIII)} \quad C_a|C_{\frac{\pi}{2}} S_d C_b \text{ or } C_b S_d C_{\frac{\pi}{2}}|C_a \quad \begin{cases} 0 \le a \le \frac{\pi}{2}\,,\ 0 \le b \le \frac{\pi}{2}\,,\ 0 \le d \\ \text{and } a+b \le \frac{\pi}{2} \text{ if } u_2 \text{ is constant} \\ \text{on every arc } C \end{cases}$$

Proof: Our method is as follows:

1. We consider a path $\mathcal{T}$ linking a point (M,θ) to the origin, such that $|\theta| > \pi$.

2. We select a part of $\mathcal{T}$ located between two configurations (M_1, θ_1) and (M_2, θ_2) such that $|\theta_1 - \theta_2| = \pi$. According to lemma 14 we replace this part by an isometric one, along which the point's orientation rotates in the opposite direction. In this way we construct a trajectory equivalent to $\mathcal{T}$ i.e having the same length and linking (M, θ) to the origin.
3. We prove that this new trajectory does not verify the necessary conditions given by PMP. As $\mathcal{T}$ is equivalent to this non optimal path we deduce that it is not optimal.

Let us consider first a type (I) path. Due to the symmetry properties it suffices to regard a path $l_a^+ l_b^- l_e^+$ with $a + b + e = \pi + \epsilon, (\epsilon > 0)$ and $a > \epsilon$. If we keep in place a piece of length ϵ and replace the final part using lemma 14, we obtain an equivalent path $l_\epsilon^+ r_e^- r_b^+ r_{a-\epsilon}^-$ which is obviously not optimal because the robot goes twice to the same configuration.

We use the same reasoning to show that a path $C_a|C_{\frac{\pi}{2}} S_d$ with $d \neq 0$ cannot be optimal if $a > \frac{\pi}{2}$. Without lost of generality we consider a path $l_{\frac{\pi}{2}+\epsilon}^+ l_{\frac{\pi}{2}}^- s_d^-$. According to lemma 14 we can replace the initial piece $l_{\frac{\pi}{2}+\epsilon}^+ l_{\frac{\pi}{2}-\epsilon}^-$ by the isometric one $r_{\frac{\pi}{2}-\epsilon}^+ r_{\frac{\pi}{2}+\epsilon}^-$. The initial path is then equivalent to the path $r_{\frac{\pi}{2}-\epsilon}^+ r_{\frac{\pi}{2}+\epsilon}^- l_\epsilon^- s^-$ which cannot be optimal as the point of inflection do not belong to the line supporting the line segment.

Consider now a path $C_a|C_{\frac{\pi}{2}} S_d C_b$ or $C_b S_d C_{\frac{\pi}{2}}|C_a$ with u_2 constant on the arcs. We show that such a path cannot be optimal if $a + b > \frac{\pi}{2}$. Consider a path $l_a^+ l_{\frac{\pi}{2}}^- s_d^- l_b^-$ with $a + b = \frac{\pi}{2} + \epsilon$ and $a > \epsilon$. We keep in place a piece of length ϵ and replace the final part by an isometric one according to lemma 14. We obtain an equivalent path $l_\epsilon^+ r_b^+ s_d^+ l_{\frac{\pi}{2}}^+ r_{a-\epsilon}^-$. As the point of inflection does not lie on the line D_0, this path violates both necessary conditions **A** and **B** of PMP (see lemma 11) and therefore is not optimal. □

Remark 9. *In the sufficient family (22) refined by lemma* 15, *the orientation of initial points is defined in* $[-\pi, \pi]$. *So, to solve the shortest path problem in* $\mathbf{R}^2 \times S^1$, *we only have to consider paths starting from* $\mathbf{R}^2 \times [-\pi, \pi]$ *in* $\mathbf{R}^3$.

Construction of domains For each type of path in the new sufficient family, we want to compute the domains of all possible starting points for paths ending at the origin. According to the symmetry properties it suffices to consider paths starting from one of the four quadrants made by Δ_θ and $\Delta_\theta^\perp$, in each plane P_θ, and only for positive or negative values of θ. We have chosen to construct domains covering the *first* quadrant (i.e. $x \tan \frac{\theta}{2} \leq y \leq -x \cot \frac{\theta}{2}$), for $\theta \in [-\pi, 0]$.

As any path in the sufficient family is described by three parameters, each domain is the image of the product of three real intervals by a continuous mapping. It follows that such domains are connected in the configuration space.

To represent the domains, we compute their restriction to planes P_θ. As θ is fixed, the cross section of the domain in P_θ is defined by two parameters. By

fixing one of them as the other one varies, we compute a foliation of this set. This method allows us, on the one hand to prove that only one path starts from each point of the corresponding domain, and on the other hand to characterize the analytic expression of boundaries.

In order to cover the first quadrant we have selected one special path for each of the nine different kinds of path of the sufficient family; by symmetry all other domains may be obtained.

In the following we construct these domains, one by one, in P_θ. For each kind of path, integrating successively the differential system on the time intervals during which (u_1, u_2) is constant, we obtain the parametric expression of initial points. In each case we obtain the analytical expression of boundaries; computations are tedious but quite easy (a more detailed proof is given in [33]).

We do not describe here the construction of all domains. We just give a detailed account of the computation of the first domain, the eight other domains are constructed exactly the same way. Figure 9 presents the covering of the first quadrant in $P_{-\frac{\pi}{4}}$, the different domains are represented.

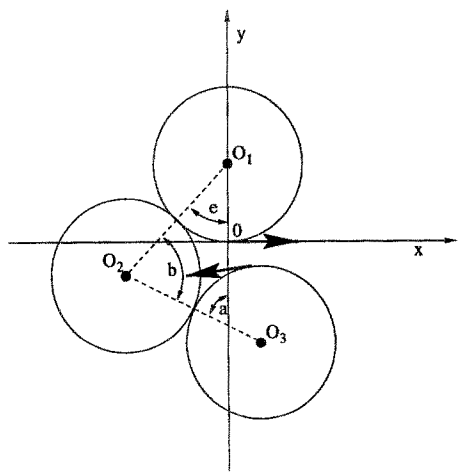


Fig. 7. Path $l_a^+ l_b^- l_e^+$

Construction of domain of path $C|C|C$: As we said in the introductive section, Sussmann and Tang have shown that the study of family $C|C|C$ may be restricted to paths types $l^+l^-l^+$ and $r^+r^-r^+$. As we only consider values of θ in $[-\pi, 0]$ it suffice to study the type $l_a^+ l_b^- l_e^+$ (figure 7). By lemma 15, a, b and e are positive real numbers verifying: $0 \leq a + b + e \leq \pi$.

Along this trajectories the control (u_1, u_2) takes successively the values $(+1,+1),(-1,+1)$ and $(+1,+1)$. By integrating the system (4) for each of these successive constant values of u_1 and u_2, from the initial configuration (x, y, θ) to the final configuration $(0,0,0)$ we get:

$$\begin{cases} x = \sin\theta + 2\sin(b+e) - 2\sin e \\ y = -\cos\theta + 2\cos(b+e) - 2\cos e + 1 \\ \theta = -a - b - e \end{cases} \tag{23}$$

Let us now consider that the value of θ is fixed. The arclength parameter e varies in $[0, -\theta]$; given a value of e, b varies in $[0, -\theta - e]$. When e is fixed as b varies, the initial point traces an arc of the circle ζ_e of radius 2 centered at $P_e(\sin\theta + 2\sin e, -\cos\theta - 2\cos e + 1)$

One end point of this arc is the point $E(\sin\theta, -\cos\theta + 1)$ (when $b = 0$), depending on the value of e the other end point (corresponding to $b = -\theta - e$) describes an arc of circle of radius 2 centered at the point $H(-\sin\theta, \cos\theta + 1)$ and delimited by the point E (when $e = -\theta$) and its symmetric F with respect to the origin O (when $e = 0$).

For different values of e the arcs of ζ_e make a foliation of the domain; this ensures the existence of a unique trajectory of this type starting form every point of the domain. Figure (8) represents this construction for two different values of θ. The cross section of this domain appears at figure (9) with the eight other domains making the covering of the first quadrant in $P_{-\frac{\pi}{4}}$.

- As this domain is symmetric about the two axes Δ_θ and $\Delta_\theta^\perp$, it follows from lemma 12 that the domain of path $l^- l^+ l^-$ is exactly the same one. This point corroborates the result by Sussmann and Tang which states that the search for an optimal path of the family $C|C|C$ (when $\theta \leq 0$) may be limited to one of these two path types.
- When $\theta = -\pi$ the domain is the disc of radius 2 centered at the origin.

Following the same method the eight other domains are easily computed (see [33]), they are represented at figure 9 in the plane $P_{-\frac{\pi}{4}}$. The domain's boundaries are piecewise smooth curves of simple sort: arcs of circle, line segments, arcs of conchoids of circle or arcs of cardioids.

Analysis of the construction As we know exactly the equations of the piecewise smooth boundary curves, we can precisely describe the domains in each plane P_θ. This construction insures the complete covering of the first quadrant, and by symmetry the covering of the whole plane. All types in the

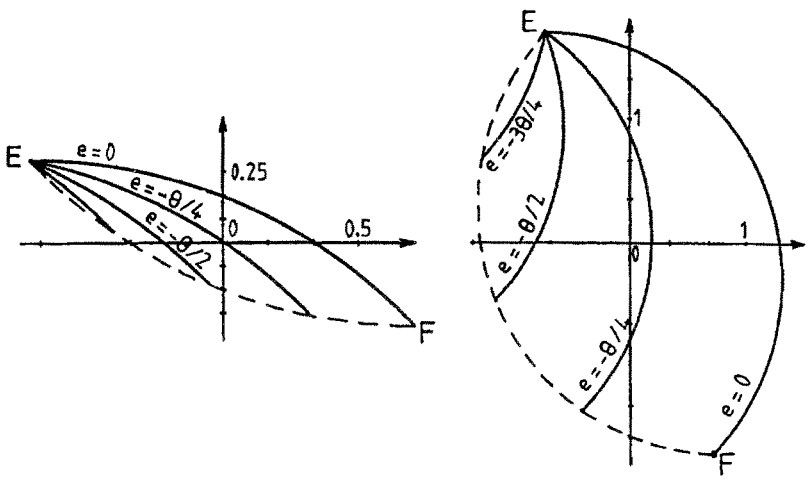


Fig. 8. Cross section of the domain of path $l_a^+ l_b^- l_e^+$ in P_θ, ($\theta = -\frac{\pi}{4}$ left side) and ($\theta = -\frac{3\pi}{4}$ right side).

sufficient family are represented[3]. Analysing the covering of the first quadrant, we can note that almost all the domains are adjacent, describing a continuous variation of the path shape. Nevertheless some domains overlap and others are not wholly contained in the first quadrant. Therefore, if we consider the covering of the whole plane (see fig 10), many intersections appear. In a region belonging to more than one domain, several paths are defined, and finding the shortest one will require a deeper study. At first sight, the analysis of all intersections seems to be combinatorially complex and tedious, but we will show that some geometric arguments may greatly simplify the problem. First, let us consider the following remarks about the domains covering the first quadrant:

- Except for the domain $r^+l^+l^-r^-$, all domains are adjacent two-by-two (i.e. they only have some parts of their boundary in common). Then, inside the first quadrant we only have to study the intersection of the domain $r^+l^+l^-r^-$ with the neighbouring domains.
- Some domains are not wholly contained in the first quadrant, therefore, they may intersect domains covering other quadrants. Nevertheless, among

[3] However, each domain is only defined for θ belonging to a subset of $[-\pi, \pi]$. So in a given plane P_θ only the domains corresponding to a subfamily of family (22) refined by lemma 15 appear.

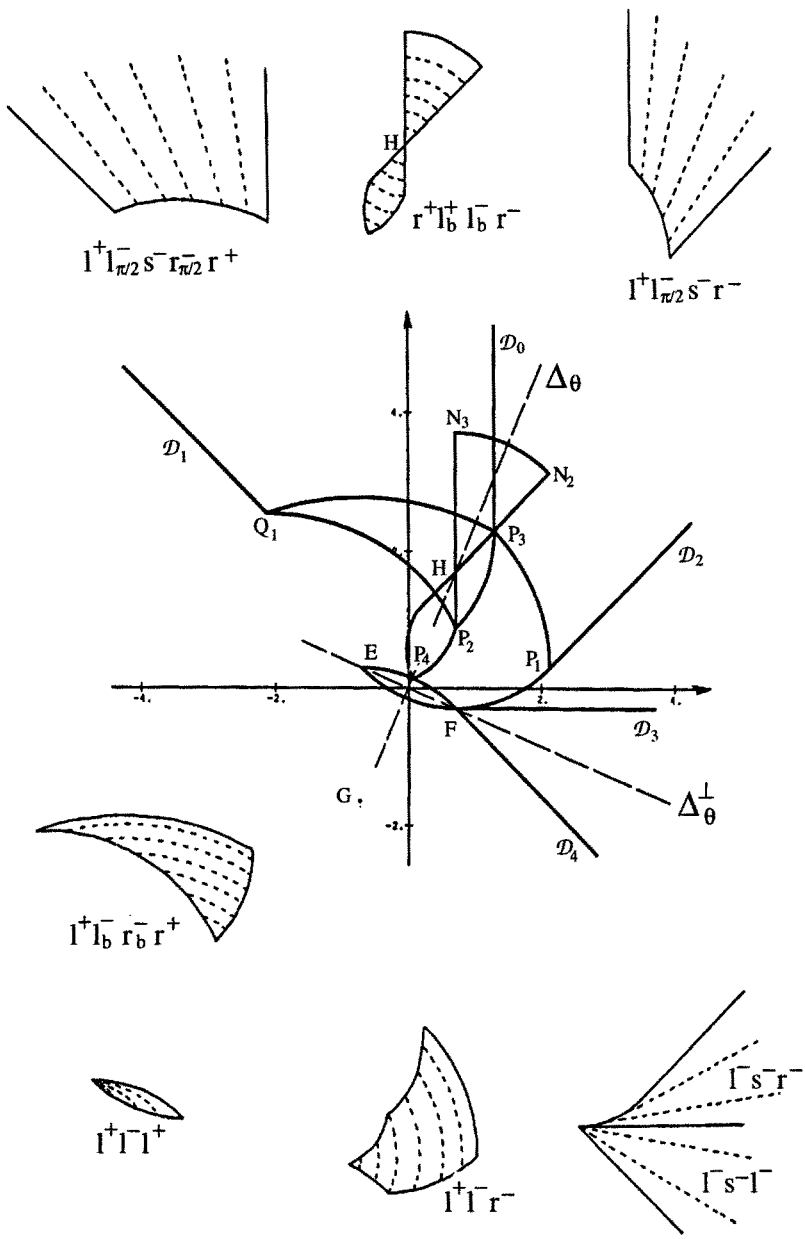


Fig. 9. The various domains covering the first quadrant in $P_{-\frac{\pi}{4}}$ (foliations appear in dotted line).

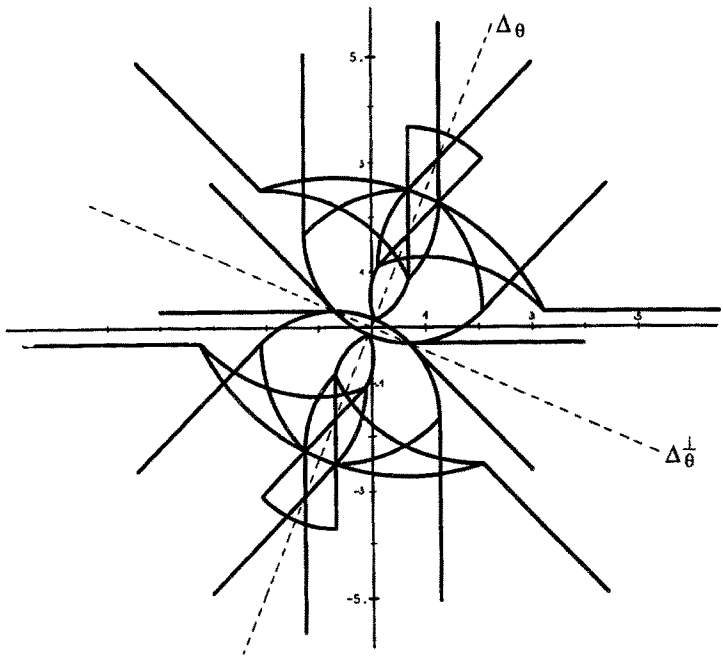


Fig. 10. Overlapping of domains covering the plane $P_{-\frac{\pi}{4}}$.

the domains overlapping other quadrants, some are symmetric about Δ_θ or $\Delta_\theta^\perp$. These domains are:

- the domains $l^+l^-l^+$ and $r^+l^+l^-r^-$ symmetric about Δ_θ,
- the domains $l^+l^-l^+$ and $l^-s^-l^-$ symmetric about $\Delta_\theta^\perp$, (i.e. all domains intersecting $\Delta_\theta^\perp$.)

In this case, we consider that only one half of the domain belongs to the covering of first quadrant. Therefore, no intersections may occur with the symmetric domains.

Finally, we only have to study two kinds of intersections: on the one hand the intersections of pairs of symmetric domains with respect to Δ_θ, (section 4.3), and on the other hand the intersections inside the first quadrant between the domain $r^+l_b^+l_b^-r^-$ and the neighbouring domains (section 4.3).

Refinement of domains intersecting Δ_θ In this section we prove that the path $l^+l^-r^-$, $l^+l_b^-r_b^-r^+$, $l^+l_{\frac{\pi}{2}}^-s^-r_{\frac{\pi}{2}}^-r^+$, and $l^+l_{\frac{\pi}{2}}^-s^-r^-$, stop being optimal as soon as their projections in P_θ cross the Δ_θ-axis. This will allow us to remove the part of these domains lying out of the first quadrant.

1/ Path $l^+l^-r^-$

Lemma 16. A path $l^+l^-r^-$ linking a directed point $(M(x,y),\theta)$ to $(0,0,0)$, with $y > -x\cot\frac{\theta}{2}$, is never optimal.

Proof: Suppose that there is a path $\mathcal{T}_1$ of type $l^+l^-r^-$ from a directed point $(M_1(x_1,y_1),\theta_1)$ to $(0,0,0)$, verifying $y_1 > -x_1\cot\frac{\theta_1}{2}$. Let M_2 be the cusp point (Figure 11). M_2 is such that [4] $y_2 < -x_2\cot\frac{\theta_2}{2}$. Let us consider a directed point (M,θ) moving along the path from (M_1,θ_1) to (M_2,θ_2). As M moves, the direction θ increases continuously from θ_1 to θ_2. As a result, the corresponding line Δ_θ varies from Δ_{θ_1} to Δ_{θ_2}. Its slope increases continuously from $-\cot\frac{\theta_1}{2}$ to $-\cot\frac{\theta_2}{2}$. Then, by continuity, there exists a directed point (M_α,α) on the arc (M_1,M_2), verifying $y_\alpha = -x_\alpha\cot\frac{\alpha}{2}$. From lemma 12, there exist two isometric paths of type $l^+l^-r^-$ and $r^+l^+l^-$ linking (M_α,α) to the origin. Thus, (M_1,θ_1) is linked to the origin by a path of type $l^+r^+l^+l^-$ having the same length as $\mathcal{T}_1$. Such a path violates both necessary conditions **A** and **B** of PMP (lemma 11): (**A:** D_0 and D^+ cannot be parallel) and (**B:** u_2 is not constant). As a consequence, $\mathcal{T}_1$ is not optimal. □

2/ Path $l^+l_{\frac{\pi}{2}}^-s^-r^-$

[4] This assertion can be easily deduced from the construction of the domain of path $l^-s^-r^-$.

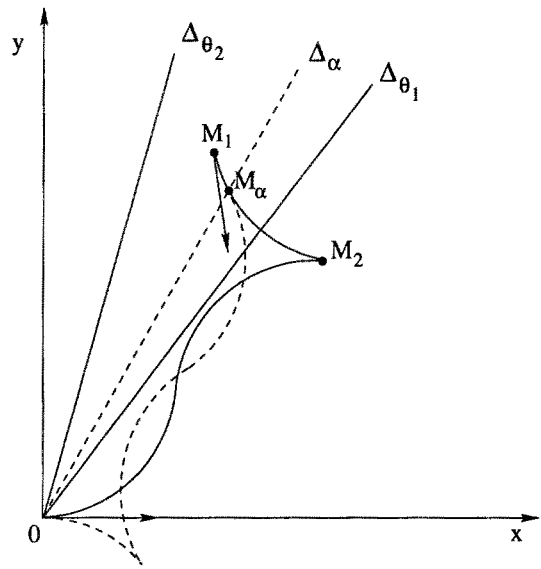


Fig. 11. There exists a point M_α such that $M_\alpha \in \Delta_\alpha$.

The shape of this path is close to the shape of the path $l_a^+ l_b^- r_c^-$ ($b = \frac{\pi}{2}$ and a line segment is inserted between the last two arcs). Then, we can use exactly the same reasoning to prove the following lemma:

Lemma 17. A path $l^+ l_{\frac{\pi}{2}}^- s r^-$ linking a directed point $(M(x,y),\theta)$ to $(0,0,0)$, with $y > -x \cot \frac{\theta}{2}$, is never optimal.

3/ Path $l^+ l_b^- r_b^- r^+$

Lemma 18. A path $l^+ l_b^- r_b^- r^+$ linking a directed point $(M(x,y),\theta)$ to $(0,0,0)$, with $y > -x \cot \frac{\theta}{2}$, is never optimal.

Proof: The reasoning is the same as in the proof of lemma 16. Assume that there is a path $\mathcal{T}_1$ of type $l^+ l_b^- r_b^- r^+$ linking a directed point $(M_1(x_1,y_1),\theta_1)$, verifying $y_1 > -x_1 \cot \frac{\theta_1}{2}$, to $(0,0,0)$. Let M_2 be the cusp-point; the subpath of $\mathcal{T}_1$ from (M_2,θ_1) to the origin is of the type $l^- r^- r^+$ symmetric to the type treated in Lemma 16. Therefore, the coordinates of M_2 must verify $y_2 < -x_2 \cot \frac{\theta_2}{2}$. Now, let us consider a directed point (M,θ) moving along the arc from (M_1,θ_1) to (M_2,θ). With the same arguments as in the proof of Lemma 16, there exists a directed point (M_α,α) on this arc, with $\theta_1 \leq \alpha \leq \theta_2$, verifying $y_\alpha = -x_\alpha \cot \frac{\alpha}{2}$. From lemma 12, there exist two isometric paths of types $l^+ l_b^- r_b^- r^+$ and $r^- r_b^+ l_b^+ l^-$ linking (M_α,α) to the origin. As a

result, (M_1, θ_1) is linked to the origin by a path of the type $l^+r^-r_b^+l_b^+l^-$ having the same length as $\mathcal{T}_1$. This path is not optimal because the robot goes twice through the same configuration; therefore $\mathcal{T}_1$ cannot be optimal. □

4/ Path $l^+l_{\frac{\pi}{2}}^-s^-r_{\frac{\pi}{2}}^-r^+$

The shape of this path is close to the shape of the path $l^+l_b^-r_b^-r^+$ ($b = -\frac{\pi}{2}$ and a line segment is inserted between the two middle arcs). Then, we can use exactly the same arguments to prove the next lemma.

Lemma 19. A path $l^+l_{\frac{\pi}{2}}^-s^-r_{\frac{\pi}{2}}^-r^+$ linking a directed point $(M(x,y),\theta)$ to $(0,0,0)$, with $y > -x\cot\frac{\theta}{2}$, is never optimal.

Now, with lemmas 16 to 19 we can remove the part of domains $l^+l^-r^-$, $l^+l_{\frac{\pi}{2}}^-s^-r^-$, $l^+l^-r^-r^+$ and $l^+l_{\frac{\pi}{2}}^-s^-r_{\frac{\pi}{2}}^-r^+$ lying out of the first quadrant (on the other side of Δ_θ). Moreover, according to the analyse made at section 4.3, we only have to consider, the half part of the domains symmetric about Δ_θ or $\Delta_\theta^\perp$ located in the first quadrant. As every domain intersecting $\Delta_\theta^\perp$ is symmetric about this axis, we can construct the covering of all other quadrants without generating new intersections. Inside each quadrant, it remains to study the intersection between the domain of path $C|C_bC_b|C$ and the neighbouring domains. Once again we restrict ourselves to the first quadrant.

Intersections inside the first quadrant From the construction of domains covering the first quadrant, it appears that the domain $r^+l_b^+l_b^-r^-$ may intersect the following three adjacent domains: $l^+l_{b'}^-r_{b'}^-r^+$, $l^+l_{\frac{\pi}{2}}^-s^-r_{\frac{\pi}{2}}^-r^+$ and $l^+l_{\frac{\pi}{2}}^-s^-r^-$. Furthermore the intersection between the domain $r^+l_b^+l_b^-r^-$ and the domains $l^+l_{\frac{\pi}{2}}^-s^-r_{\frac{\pi}{2}}^-r^+$ and $l^+l_{\frac{\pi}{2}}^-s^-r^-$ only happens when b is strictly greater than $\frac{\pi}{3}$. First, as a corollary of lemma 16, we are going to prove that a path $r^+l_b^+l_b^-r^-$ is never optimal when $b > \frac{\pi}{3}$. Therefore the corresponding part of this domain will be removed and the intersections of domains inside the first quadrant will be reduced to the overlapping of domains $r^+l_b^+l_b^-r^-$ and $l^+l_{b'}^-r_{b'}^-r^+$.

Corollary 1. A path of the family $CC_b|C_bC$ verifying $b > \frac{\pi}{3}$ cannot be optimal.

Proof: Let us consider a path of the type $r_a^+l_b^+l_b^-r_e^-$. If this path is optimal, then the subpath $l_b^+l_b^-r_e^-$ is also optimal. Integrating the corresponding system we obtain the expression of initial points coordinates:

$$\begin{cases} x = \sin\theta - 2\sin(e-b) + 2\sin e \\ y = -\cos\theta + 2\cos(e-b) - 2\cos e + 1 \end{cases}$$

with $\theta = e - 2b$ (since the first two arcs of circles have the same length) and from lemma 16 the coordinates must verify $y \leq -\cot\frac{\theta}{2}x$. Replacing x and y by their parametric expression, we obtain after computation:

$$\sin\frac{e}{2}(2\cos b - 1) \geq 0 \quad \text{then} \quad b \leq \frac{\pi}{3} \quad (\text{since} \quad 0 < e \leq \frac{\pi}{2}) \quad \square$$

Therefore according to the previous construction we may remove the part of the domain $r^+l_b^+l_b^-r^-$ located beyond the point H with respect to O. (see figure 9).

Now, only one intersection remains inside the first quadrant, between the domains $r_a^+l_b^+l_b^-r_e^-$ and $l_{a'}^+l_{b'}^-r_{b'}^-r_{e'}^+$; let us call $\mathcal{I}$ this region. In order to determine which paths are optimal in this region, we compute in each plane P_θ the set of points that may be linked to the origin by a path of each kind having the same length. Initial point of these two paths are respectively defined by the following parametric systems:

$$(r_a^+l_b^+l_b^-r_e^-) \begin{cases} x = -\sin\theta + 2(2\cos b - 1)\sin(e-b) \\ y = \cos\theta - 2(2\cos b - 1)\cos(e-b) + 1 \end{cases}$$

$$(l_{a'}^+l_{b'}^-r_{b'}^-r_{e'}^+) \begin{cases} x = \sin\theta - 4\sin e' + 2\sin(e'+b') \\ y = -\cos\theta + 4\cos e' - 2\cos(e'+b') - 1 \end{cases} \tag{24}$$

the length of these paths are respectively:

$$\begin{cases} L = a + 2b + e = 4b + \theta \text{ with } \theta = e - 2b + a \\ L' = e' + 2b' + a' = 2(b' + e') - \theta \text{ with } \theta = e' - a' \end{cases} \tag{25}$$

The required condition $L = L'$ implies that $\theta + 2b - b' - e' = 0$. By replacing $e' + b'$ by $\theta + 2b$ in the second system, then writing that both systems are equivalent we obtain:

$$\begin{cases} \sin(e-b)(1-2\cos b) + \sin\theta - 2\sin e' + \sin(\theta + 2b) = 0 \\ \cos(e-b)(1-2\cos b) + \cos\theta - 2\cos e' + \cos(\theta + 2b) + 1 = 0 \end{cases}$$

we eliminate the parameter e' writing that $\sin^2(e') + \cos^2(e') = 1$; then after computation, we obtain the following relation between e and b:

$$4\cos^2 b - 2\cos b + (1 - 2\cos b)(2\cos(e - 2b - \theta)\cos b + \cos(e-b)) + \cos\theta + \cos(\theta + 2b) - 1 = 0 \tag{26}$$

As $e-2b-\theta = e-b-(b+\theta)$, this equation may be rewritten as follows:

$$A\sin(e-b) + B\cos(e-b) + C = 0$$

where A, B and C are functions of b and θ defined by:

$$\begin{cases} A = 2(1-2\cos b)\sin(b+\theta)\cos b \\ B = (1-2\cos b)(2\cos(b+\theta)\cos b + 1) \\ C = 4\cos^2 b - 2\cos b + \cos\theta + \cos(\theta+2b) - 1 \end{cases}$$

Therefore we can express $\sin(e-b)$ and $\cos(e-b)$ by solving a second degree equation; we obtain:

$$\sin(e-b) = \frac{-AC \pm |B|\sqrt{A^2+B^2-C^2}}{A^2+B^2} \tag{27}$$

The discriminant $\mathcal{D} = A^2 + B^2 - C^2$ may be factored as follows:

$$\mathcal{D} = 4\cos(b)\sin^2(\frac{b+\theta}{2})(\cos(2b+\theta) + \cos(\theta) + 6\cos(b) - 4)$$

therefore, as $b \in [0, \frac{\pi}{3}]$, the sign of $\mathcal{D}$ is equal to the sign of

$$E(b) = \cos(2b+\theta) + \cos(\theta) + 6\cos(b) - 4$$

Let us call $b_{\max}$ the value of b solution of $E(b) = 0$. As E(b) is a decreasing function of b, E(b) is positive when $b \leq b_{\max}$. We will see later that the maximal value of b we have to consider verifies this condition, ensuring our problem to be well defined.

Now, as the region $\mathcal{I}$ is delimited by the vertical line (P_2, N_3), each point belonging to $\mathcal{I}$ must verify: $x \leq \sin\theta$. Moreover, as the type $r^+l_b^+l_b^-r^-$ is defined for $b \in [-\theta, \frac{\pi}{3}]$, we can deduce from the first line of system (24) that $\sin(e-b)$ is negative. As a result, the choice of the positive value of the discriminant in (27) cannot be a solution of our problem. From this condition we determine a unique expression for $\sin(e-b)$ and $\cos(e-b)$. Replacing these expressions in system (24) we obtain the parametric equation of a curve γ_θ issued from P_2 (when $b = -\theta$), dividing the region $\mathcal{I}$ into two subdomains, and crossing the axis Δ_θ at a point T (see figure 12). The value b_T of b corresponding to the point T may be characterized in the following manner:

From lemma 12 we know that any path of type $r_a^+l_b^+l_b^-r_e^-$ starting on Δ_θ verifies $a = e$; it follows that $e - b - \frac{\theta}{2} = 0$. Replacing $e-b$ by $\frac{\theta}{2}$ in (26) b_T appears as being the solution of the implicit relation $R(b) = 0$, where:

$$R(b) = 4\cos^2 b - 2\cos b + (1 - 2\cos b)(2\cos(b + \frac{\theta}{2})\cos b + \cos\frac{\theta}{2}) + \cos\theta + \cos(\theta + 2b) - 1 \quad (28)$$

Now, combining the relation $R(b_T) = 0$ with the expression of $E(b_T)$ we can prove simply that $b_T \leq b_{\max}$ in order to insure the sense of our result. Therefore, R(b) being a decreasing function of b, the values of $b \in [-\theta, b_T]$ are the values of $b \in [-\theta, \frac{\pi}{3}]$ verifying $R(b) \geq 0$. The curve γ_θ is the only set of points in $\mathcal{I}$ where both paths have the same length. As the distance induced by the shortest path is a continuous function of the state, this curve is the real limit of optimality between these two domains. This last construction achieves the partition of the first quadrant, and by the way the partition of the whole plane.

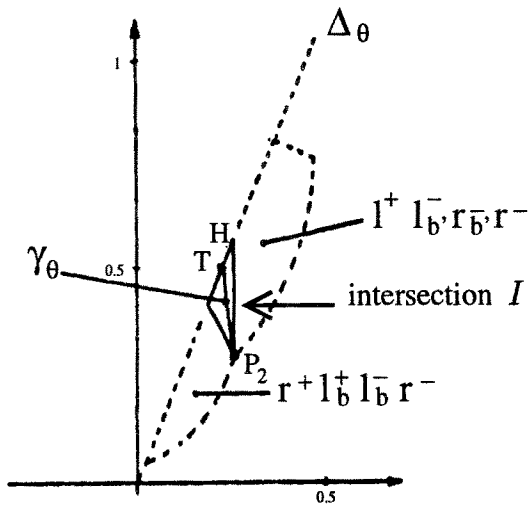
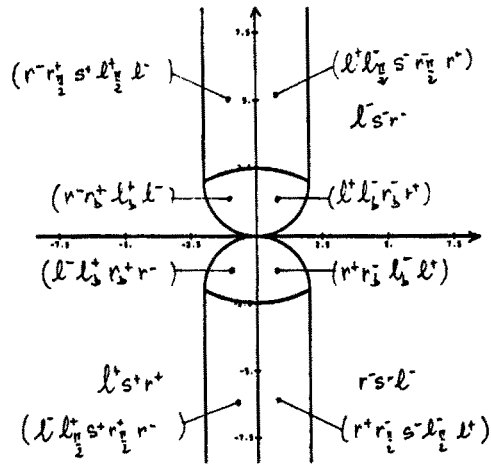
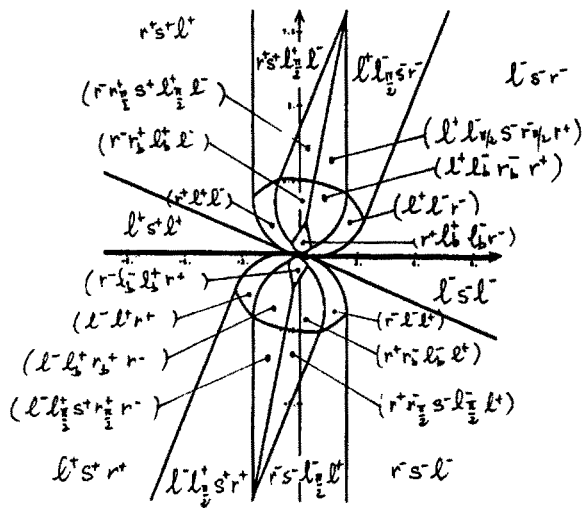


Fig. 12. The curve γ_θ splitting the intersection of domains $r^+ l_b^+ l_b^- r^-$ and $l^+ l_{b'}^- r_{b'}^- r^-$ in the first quadrant of $P_{-\frac{\pi}{4}}$

Description of the partition Figures (13),(14) and (15) show the partition of the plane P_θ for several values of θ all these pictures have been traced from the



$\theta = 0$



$\theta = -\frac{\pi}{8}$

Fig. 13. Partitions of planes P_0 and $P_{-\frac{\pi}{8}}$

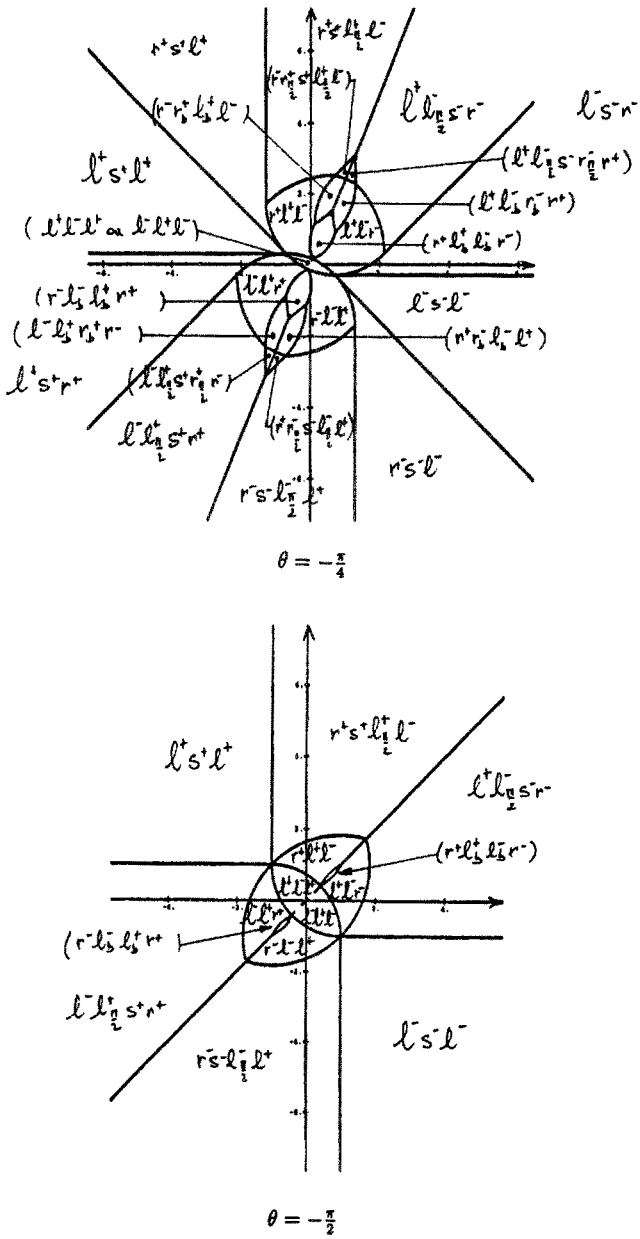


Fig. 14. Partitions of planes $P_{-\frac{\pi}{4}}$ and $P_{-\frac{\pi}{2}}$

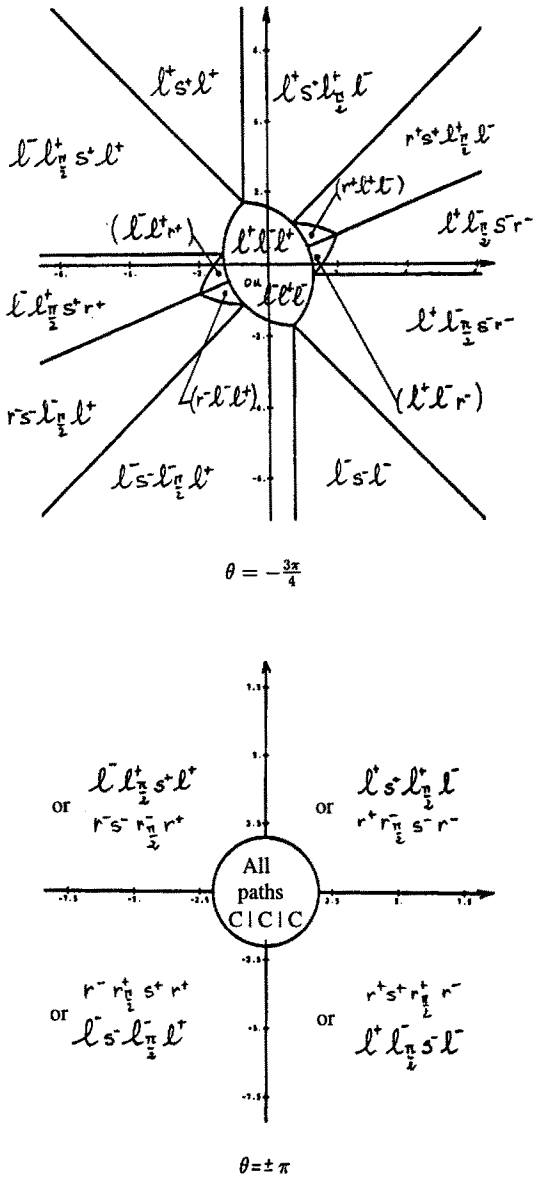


Fig. 15. Partitions of planes $P_{-\frac{3\pi}{4}}$ and $P_{\pm\pi}$

analytical equations of boundaries by using the symbolic computation software *Mathematica*. Each elementary cell consists of directed points that may be linked to the origin by the same kind of optimal path. The 46 domains never appear together in a plane P_θ; the following table presents the values of θ for which each domain exists[5]

Type	*Intervals of validity*
$CC_u\|C_uC$	$[-\frac{2\pi}{3}, 0]$ and $[0, \frac{2\pi}{3}]$
$C\|C_uC_u\|C$	$[-\frac{\pi}{3}, \frac{\pi}{3}]$
$C\|CC$ and $CC\|C$	$[-\pi, 0]$ and $[0, \pi]$
$C\|C_{\frac{\pi}{2}}SC$ and $CSC_{\frac{\pi}{2}}\|C$ if $sign(u_2)$ changes	$[-\pi, 0]$ and $[0, \pi]$
$C\|C_{\frac{\pi}{2}}SC$ and $CSC_{\frac{\pi}{2}}\|C$ if $sign(u_2)$ is constant	$[-\pi, -\frac{\pi}{2}]$ and $[\frac{\pi}{2}, \pi]$
$C\|C\|C$	$[-\pi, 0]$ and $[0, \pi]$
CSC if $sign(u_2)$ changes	$[-\frac{\pi}{2}, \frac{\pi}{2}]$
CSC if $sign(u_2)$ is constant	$[-\pi, 0]$ and $[0, \pi]$
$C\|C_{\frac{\pi}{2}}SC_{\frac{\pi}{2}}\|C$	$[-2\,\mathrm{arccot}(2), 2\,\mathrm{arccot}(2)]$

When θ varies in $[-\pi, \pi]$ the partition of planes P_θ induces a partition of $\mathbf{R}^2 \times [-\pi, \pi]$. Identifying the planes $P_{-\pi}$ and P_π we obtain a partition of the configuration space $\mathbf{R}^2 \times S^1$.

In most part of domains the optimal solution is uniquely determined. However, there exist some regions of the space where several equivalent path are defined. To describe these regions we introduce the following notation.

In the first quadrant of each plane P_θ, we denote by Δ_θ^T the half-line defined as the part of Δ_θ located beyond the point T (with respect to O). According to lemmas 16 to 19, paths $l^+l^-r^-$, $l^+l^-_{\frac{\pi}{2}}s^-r^-$, $l^+l^-_br^-_br^+$, and $l^+l^-_{\frac{\pi}{2}}s^-r^-_{\frac{\pi}{2}}r^+$ stop being optimal as soon as they cross Δ_θ^T, but are still optimal on Δ_θ^T. As the same reasoning holds for the domains symmetric with respect to Δ_θ, there exist two equivalent paths optimal for linking any point of Δ_θ^T to the origin. The same phenomenon occurs on the curve γ_θ where paths $r^+l^+_bl^-_br^-$ and $l^+l^-_br^-_br^+$ have the same length. Hence, in the first quadrant, two equivalent paths are defined at each point of $\Delta_\theta^T \cup \gamma_\theta$. By symmetry with respect to Δ_θ and $\Delta_\theta^\perp$, we can define such a set inside the four other quadrants. Let us call N_θ the union of these four symmetric sets.

[5] These values of θ have been deduced from the bounds on the parameters given by the partition. Details are given in [34].

At any point of $P_\theta \setminus N_\theta = \{p \in P_\theta,\ p \notin N_\theta\}$ a unique path is defined if $\theta \not\equiv \pi\ mod(2\pi)$. Inside each domain the uniqueness is proven by the existence of a foliation, and on the boundaries (outside N_θ) any path is defined as a continuous transition between two types (and belongs to both path types). However, according to lemma 14, when $\theta \equiv \pi\ mod(2\pi)$, two equivalent (isometric) paths are defined at any point of $P_\theta \setminus N_\theta$ and therefore, four equivalent paths are defined at any point of N_θ. As we have seen in the construction, there always exist two equivalent paths ($l^+l^-l^+$ and $l^-l^+l^-$ when $\theta > 0$) and ($r^+r^-r^+$ and $r^-r^+r^-$ when $\theta < 0$) linking any point of the central domain $C|C|C$ to the origin. Furthermore, when the initial orientation θ equals $\pm\pi$, there exist two equivalent strategies for linking any point of the plane to the origin, each one corresponding to a different direction of rotation of the point (see lemma 14). In that case each of the four paths $C|C|C$ is optimal in the central disc of radius 2.

By choosing one particular solution in each region where several optimal path are defined, one can determine a synthesis of optimal paths according to definition 7. Therefore, the determination of such a synthesis is not unique.

In each cross section P_θ, the synthesis provides a complete analytic description of the boundary of domains which appear to be of simple sort: line segment, arc of circle, arc of cardioid of circle, etc. Therefore, to characterize an optimal control law for steering a point to the origin, it suffice to determine in which cell the point is located, without having to do further test. This provides a complete solution to Reeds and Shepp's problem.

On the other hand, this study constitutes an interesting way to focus on the insufficiency of a local method, such as Pontriagyn's maximum principle, for solving this kind of problem. The Δ_θ axis appeared as a boundary and we had to remove the piece of domains lying on one side of this axis. More precisely, we have shown that any trajectory stops being optimal as it crosses the set N_θ. This phenomenon is due to the existence of several wavefronts intersecting each other on this set. For this reason two equivalent paths are defined at each point of N_θ. (each of them corresponds to a different wave front). PMP is a local reasoning based on the comparison of each trajectory with the trajectories obtained by infinitesimally perturbating the control law at each time. As this reasoning cannot be of some help to compare trajectories belonging to different wave fronts, it is necessary to use a geometric method to conclude the study, as we did in section 4.3 and 4.3. The main problem remains to determine *a priori* the locus of points where different wave fronts intersect.

The construction we have done for determining a partition of the phase space required a complex geometric reasoning. In the following section we will show how Boltianskii's verification theorem can be applied *a posteriori* to provide a simple new proof of this result.

4.4 An example of regular synthesis

In this section, we prove that the previous partition effects a regular synthesis in any open neighbourhood of O in $\mathbf{R}^2 \times S^1$. First of all, we need to prove that the curves and surfaces making up the partition define piecewise smooth sets. From the previous construction we know that the restriction of any domain to planes P_θ is a connected region delimited by a piecewise smooth boundary curve. Except for the curve γ_θ (computed at section 4.3) each smooth component $C_i(\theta)$ of the boundary remains a part of a same geometric figure $\mathcal{F}$ (line, circle, conchoid of circle, ...) as θ varies. Let $M^i(\theta)$ and $N^i(\theta)$ be the extremities of the curve $C_i(\theta)$. As θ varies, the position and orientation of $\mathcal{F}$ as well as the coordinates of $M^i(\theta)$ and $N^i(\theta)$ vary as smooth functions of θ. Therefore, in $\mathbf{R}^2 \times [-\pi, \pi]$, the lines trace smooth ruled surfaces, and the circles and conchoids draw smooth surfaces. In each case we have verified that the boundary curves $M^i(\theta)$ and $N^i(\theta)$ of these surfaces never connect tangentially, making sure that all these surfaces are non singular 2-dimensional smooth surfaces.

The study of the surface Γ, made up by the union of the horizontal curves γ_θ when θ varies in $[-\frac{\pi}{3}, \frac{\pi}{3}]$ requires more attention. As γ_θ is the region of P_θ where paths $r_a^+ l_b^+ l_b^- r_e^-$ and $l_a^+ l_b^- r_b^- r_e^+$ have the same length, the surface Γ is defined as the image of the set $D_\Gamma = \{(\theta, b) \in \mathbf{R}^2, \theta \in [-\frac{\pi}{3}, 0], b \in [-\theta, b_T]\}$ by the following mapping:

$$\begin{cases} x = -\sin\theta + 2(2\cos b - 1)\sin(e - b) \\ y = \cos\theta - 2(2\cos b - 1)\cos(e - b) + 1 \\ \theta = e - 2b + a \end{cases}$$

where $\sin(e - b)$ and $\cos(e - b)$ are deduced from formula (27) and b_T is the solution of the implicit equation $R(b) = 0$ where $R(b)$ is given by (28).

Using the symbolic computation software *Mathematica* we have checked that the matrix of partial derivatives has full rank 2 at each point of D_Γ. Therefore, as the domain D_Γ is a 2-dimensional region of the plane (θ, b) without singularities, delimited by two smooth curves, Γ constitutes a 2-dimensional smooth surface (see figure 16).

All the pieces of surfaces, making up the partition are 2-dimensional smooth surfaces and from remark 4 we know that they constitute 2-dimensional piecewise-smooth sets. If P^2 is the union of these surfaces, P^1 the union of their smooth boundary curves $M^i(\theta)$ and $N^i(\theta)$, P^0 the target point O, then in any open neighbourhood $\mathcal{V}$ of O we can write the required relation: $P^0 \subset P^1 \subset P^2 \subset \mathcal{V}$.

In order to check the regularity conditions we have considered each trajectory one-by-one. following the representative point from the initial point to the origin we have analysed the different cells encountered. We have checked that the cell's dimension varies according to the hypothesis **B** of definition 9. In each

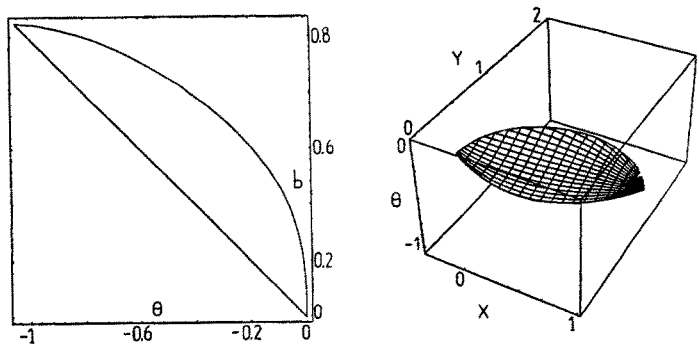


Fig. 16. Set D_Γ (left) and surface Γ (right)

case we have verified that the point never reaches the next cell tangentially. For each trajectory we can represent this study within a table by describing from the top to the bottom the cells σ_i successively crossed. Each cell corresponds to a subpath type represented by a subword of the initial word. In each case we specify the dimension and the type (T_1) or (T_2) of the cells encountered. When the point passes from a cell σ_i to a cell $\Pi(\sigma_i) = \sigma_{i+1}$ we verify that the trajectory riches σ_{i+1} with a nonzero angle α_i. This is done by comparing the vector $\boldsymbol{v}_i$ tangent to the trajectory with a vector $\boldsymbol{n}_{i+1}$ normal to σ_{i+1} (if σ_{i+1} is a 2-dim cell), or with a vector $\boldsymbol{w}_{i+1}$ tangent to σ_{i+1} (if σ_{i+1} is a 1-dim cell). In any case the last cell, described in the bottom of the table, is a 1-dimensional cell which is a piece of trajectory linking the point to P_0.

Due to the lack of place we just present here the table corresponding to paths $l_a^+ l_{\frac{\pi}{2}}^- s_d^- r_{\frac{\pi}{2}}^- r_e^+$, an exhaustive description of all path types may be found in [35].

Path $l_a^+ l_{\frac{\pi}{2}}^- s_d^- r_{\frac{\pi}{2}}^- r_e^+$					
Cell	Dim	Type	$\boldsymbol{v}_i$	$\boldsymbol{n}_i$ or $\boldsymbol{w}_i$	angle α_i
$\sigma_1 : l_a^+ l_{\frac{\pi}{2}}^- s_d^- r_{\frac{\pi}{2}}^- r_e^+$	3	T_1	$\boldsymbol{v}_1 \begin{pmatrix} \cos\theta \\ \sin\theta \\ 1 \end{pmatrix}$		
$\sigma_2 : l_{\frac{\pi}{2}}^- s_d^- r_{\frac{\pi}{2}}^- r_e^+$	2	T_2		$\boldsymbol{n}_2 \begin{pmatrix} \cos\theta \\ \sin\theta \\ 3+d \end{pmatrix}$	$\boldsymbol{n}_2.\boldsymbol{v}_1 = d+4 \neq 0$ because $d > 0$ then $\alpha_2 \neq 0$
$\sigma_3 : l_b^- s_d^- r_{\frac{\pi}{2}}^- r_e^+$	3	T_1	$\boldsymbol{v}_3 \begin{pmatrix} -\cos\theta \\ -\sin\theta \\ 1 \end{pmatrix}$		
$\sigma_4 : s_d^- r_{\frac{\pi}{2}}^- r_e^+$	2	T_1	$\boldsymbol{v}_4 \begin{pmatrix} -\cos\theta \\ -\sin\theta \\ 0 \end{pmatrix}$	$\boldsymbol{n}_4 \begin{pmatrix} \sin\theta \\ -\cos\theta \\ 2+d \end{pmatrix}$	$\boldsymbol{n}_4.\boldsymbol{v}_3 = 2+d \neq 0$ because $d > 0$ then $\alpha_4 \neq 0$
$\sigma_5 : r_{\frac{\pi}{2}}^- r_e^+$	1	T_2		$\boldsymbol{w}_5 \begin{pmatrix} \cos\theta - 2\sin\theta \\ \sin\theta + 2\cos\theta \\ 1 \end{pmatrix}$	$\boldsymbol{v}_4$ and $\boldsymbol{w}_5$ not colinear then $\alpha_5 \neq 0$
$\sigma_6 : r_k^- r_e^+$	2	T_1	$\boldsymbol{v}_6 \begin{pmatrix} -\cos\theta \\ -\sin\theta \\ -1 \end{pmatrix}$		
$\sigma_7 : r_e^+$	1	T_1		$\boldsymbol{w}_7 \begin{pmatrix} \cos\theta \\ \sin\theta \\ -1 \end{pmatrix}$	$\boldsymbol{v}_6$ and $\boldsymbol{w}_7$ not colinear then $\alpha_7 \neq 0$

Now, let us analyse carefully the other regularity conditions: Let N be the set defined by $N = \cup_{\theta \in [-\pi,\pi]} N_\theta$ where N_θ is the set defined at section 4.3 as the union of γ_θ, Δ_θ^T and their image by the axial symmetries with respect to Δ_θ and $\Delta_\theta^\perp$. From the previous reasoning we know that N is a piecewise smooth set. Let v be the function defined in $\mathcal{V}$, taking its values in the control set $U = \{(u_1, u_2), |u_1| = 1, \text{and } u_2 \in [-1, 1]\}$ which defines an optimal control law at each point. In each cell where more than one optimal solution exists the choice of a constant control has been done in order to define the function v in a unique way.

A - As stated in the beginning of this section, all the i-dim cells are i-dimensional smooth manifolds. Moreover, as each cell corresponds to a same path type, the control function v takes a constant value at each point of the cell. Therefore, v is obviously continuously differentiable inside each cell, and may be prolonged into an other constant function when the point reaches the next cell.

B - All the 3-dim cells are of type T_1

- When the representative point passes from a cell to another it never arrives tangentially. Furthermore, as $u_1 = 1$, the velocity never vanishes.
- Along the trajectory, the variation of cell types (T_1) or (T_2) follows the rule stated by Boltianskii.

C - Along any trajectory, the representative point pierces at most three cells of type T_2 and reaches the point O after a finite time.

D - From every point of N there start two trajectories having the same length and from any point of $\mathcal{V} \setminus N$ there issues a unique trajectory.

E - All these trajectories satisfy the necessary conditions of PMP.

F - By crossing a border (except the set N) from a domain to another, either the length of one elementary piece making up the trajectory vanishes, or a new piece appears. When the point crosses the set N, the optimal strategy switches suddenly for an isometric trajectory. Therefore, in any case, the path length is a continuous function of the state in $\mathcal{V}$ (see [26] for more details).

With this conditions the function v and the sets P_i effect a regular synthesis in $\mathcal{V}$. As the point moves with a constant velocity, it is equivalent to minimize the path length or the time, we have $f^0(x,u) \equiv 1$. Finally, as the coordinate functions $f^1(x,u) = \cos\theta u_1$, $f^2(x,u) = \sin\theta u_1$ and $f^3(x,u) = u_2$ have continuous partial derivatives in x and u, the hypotheses of theorem 6 are verified providing a new proof of our preceding result.

To our knowledge, this construction constitutes the first example of a regular synthesis for a nonholonomic system in a 3-dimensional space.

5 Shortest paths for Dubins' Car

Let us now present more succinctly the construction of a synthesis of optimal paths for Dubins' problem (DU). This results is the fruit of a collaboration between the project *Prisme* of INRIA Sophia Antipolis and the group *Robotics and Artificial Intelligence* of LAAS-CNRS see [10] for more details.

At first sight, this problem might appear as a subproblem of RS. Nevertheless, the lack of symmetry of the system, due to the impossibility for the car to move backwards, induces strong new difficulties. Nevertheless, the method we use for solving this problem is very close to the one developed in the preceding section.

The work is based on the sufficient family of trajectories determined by Dubins (14). Note that this sufficient family can also be derived from PMP (see [36]). The study is organized as before. First, we determine the symmetry properties of the system and we use them to reduce the state space and to refine Dubins' sufficient family. Then, in a second time we construct the domains

corresponding to each path type and we analyse their intersections. As we did in studying the problem RS, we consider the restriction of domains to planes P_θ where the orientation θ is constant.

Remark 10. *As Dubins' car only moves forwards its more convenient to fix the initial configuration of the car to be at the origin $(O,0)$ of the space, and to search for the configuration (M,θ) reachable from this point.*

5.1 Symmetry and reduction properties

As the linear velocity u_1 is fixed to 1 we can rewrite system 2 as follows:

$$\begin{cases} \dot{x} = \cos\theta \\ \dot{y} = \sin\theta \\ \dot{\theta} = u \end{cases} \tag{29}$$

where $u \in [-1,1]$ represents the angular velocity. In the study of Reeds and Shepp's problem we have shown that it was possible to construct several isometric trajectories by using simple geometric arguments. Nevertheless, as system (29) is no more symmetric, these properties are not valid for Dubins' problem. In particular, if $\mathcal{T}$ is a trajectory admissible for DU, the trajectory symmetric to $\mathcal{T}$ with respect to the point O is no more admissible. Therefore, the sole symmetry property that remains valid for DU, is the existence of isometric trajectories ending at points symmetric with respect to $\Delta_\theta^\perp$ in each plane P_θ. This result can be easily proven by using the same reasoning as the one developed in the proof of lemma 12. We use the notations introduced for the study of Reeds and Shepp's problem.

Lemma 20. In the plane of the car's motion (O,x,y) let (M,θ) be a configuration of the car and M^3 the point symmetric to M with respect to $\Delta_\theta^\perp$. If $\mathcal{T}$ is a trajectory admissible for DU starting at the origin $(O,0)$ and ending at (M,θ), there exists another admissible trajectory $\mathcal{T}^3$ isometric to $\mathcal{T}$ which links the origin to the configuration (M^3,θ).

As for RS the word describing $\mathcal{T}^3$ is obtained by reversing the word describing $\mathcal{T}$. On the other hand, the symmetry with respect to the x-axis provides another isometric admissible trajectory as follows:

Lemma 21. If $\mathcal{T}$ is an admissible trajectory for DU, starting at the origin and ending at $(M(x,y),\theta)$, there exists another admissible trajectory $\overline{\mathcal{T}}$ isometric to $\mathcal{T}$, which starts at the origin and ends at $(\overline{M}(x,-y),-\theta)$.

Dubins' sufficient family (14) contains two path types:

- $C_a S_d C_e$ with $a, e \in [0, 2\pi[$ and $d \geq 0$,
- $C_a C_b C_e$ with $a, e \in [0, 2\pi[$ and $b \in]\pi, 2\pi[$.

From lemma 20, we can restrict our study to paths: *lrl*, *rlr*, *rsr*, *lsl* and either *rsl* or *lsr*. Furthermore thanks to lemma 21 we only have to consider the values of θ such that a representative of their class modulo 2π belongs to $[0, \pi]$. Let us now state three lemmas providing additional necessary optimality conditions.

Lemma 22. A necessary condition for a path $C_a C_b C_e$ to be optimal is that:

$$\begin{cases} \pi < b < 2\pi \\ 0 \leq a \leq b \text{ and } 0 \leq e \leq b \\ 0 \leq a < b - \pi \text{ or } 0 \leq e < b - \pi \end{cases}$$

Proof: The first condition on b has been already given by Dubins [16] or in [2,36]. A characteristic straight line D_0 is defined for each optimal path (as in lemma 11), which supports line segments and where inflection points occur. On one side of this line the path turns clockwise, and on the other side, counterclockwise. Thus if $a > b$ (resp. $e > b$), the first (resp. last) arc must cross the line D_0; this is not possible.

For the last condition, suppose that the contrary is true: $a \geq b - \pi$ and $e \geq b - \pi$. Consider the circle tangent to both extremal arcs. Tracing an arc of this new circle we can build a shorter path as follows (see figure 17): an arc shortened to length $a - b + \pi$ on the first circle, concatenated to an arc of length $2\pi - b$ on the new circle followed by an arc of length $e - b + \pi$ on the last circle.□

Lemma 23. Paths *rsr* (*resp. lsl*) such that the sum of the length of the two arcs of circle is equal to 2π can be replaced by an isometric path *lsl* (*resp. rsr*).

Proof: It suffice to consider figure 18. Whenever there exists a path of type $r_a s_d r_e$ (resp. $l_a s_d l_e$) with $a + e = 2\pi$ there also exists an equivalent path $l_e s_d l_a$ (resp. $r_e s_d r_a$) □

Lemma 24. Along any optimal trajectory the maximal variation of θ is 2π.

Proof:

- Types *rsl* and *lsr*: As the directions of rotation on each arcs are opposite and the length of each arc is lower or equal to 2π, the result follows.
- From lemma 22, on types $r_a l_b r_e$ and $l_a r_b l_e$ the arclength verify: $0 \leq a \leq b$ and $0 \leq e \leq b$. therefore $|a - e + b| \leq v < 2\pi$.
- Types *rsr* and *lsl*: Suppose that the initial and final orientations are equal. From lemma 23 we know that in the case that $a + e = 2\pi$, if a path of type $r_a s_d r_e$ (resp. $l_a s_d l_e$) exists, there also exists an equivalent path of type $l_e s_d l_a$ (resp. $r_e s_d r_a$). It follows that a path $r_a s_d r_e$ (resp. $l_a s_d l_e$) with $a + e = 2\pi + \epsilon$ cannot be optimal because it is equivalent to a path $l_e s_d l_a r_\epsilon$ which does not verify the necessary conditions of PMP (points of inflection and line segment must belong to the same line D_0). □

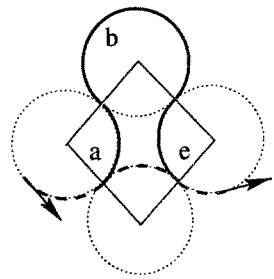


Fig. 17. Non optimal $C_a C_b C_e$ trajectory

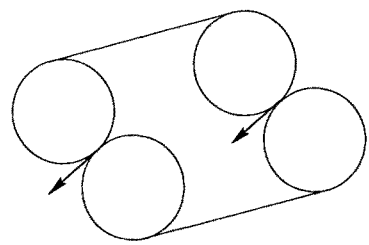


Fig. 18. Simultaneous existence of path $r_a s r_e$ and $l_e s l_a$ when $a + e = 2\pi$

Now, taking into account these new bounds on arclength, and the symmetry properties we construct the domains corresponding to each path type in planes P_θ.

5.2 Construction of domains

From lemma 20 it suffice to construct the domains of paths lsl, rsr, lsr, rlr and lrl. The domain of path rsl will be obtained from the domain of path lsr by symmetry with respect to the $\Delta_\theta^\perp$-axis.

From lemma 24 we know that the final orientation $\theta \in S^1$ may be viewed as a real number belonging to $[-2\pi, 2\pi]$. Therefore, according to lemma 21 and lemma 23, we only have to consider the cross sections of domains belonging to planes P_θ with $\theta \in [0, \pi]$ or $\theta + 2\pi \in [0, \pi]$.

Integrating the differential system (29) for the successive constant values of the input u (as we did for RS in section 4.3) we compute the cross section of each domain. We do not give here the detail of the construction (see [11]). To describe the construction we need to introduce the following notations.

* E is the point of coordinates $(\sin\theta, 1 - \cos\theta)$,
* G is the point of coordinates $(\sin\theta, -1 - \cos\theta)$,

* F (*resp.* H) is the point symmetric to E (*resp.* G) w.r.t. the origin O,
* J (*resp.* K) is the point symmetric to H (*resp.* G) w.r.t. point E,
* $\mathcal{D}_0$ (*resp.* $\mathcal{D}_1$) is the ray from E towards positive x-coordinates, of orientation 0 (*resp.* θ),
* $\mathcal{D}_2$ (*resp.* $\mathcal{D}_3$) is the ray from F parallel to $\mathcal{D}_0$ (*resp.* $\mathcal{D}_1$),
* we denote by $\mathcal{C}_P$ the circle (or the disc) centered at the generic point P and with radius 2.

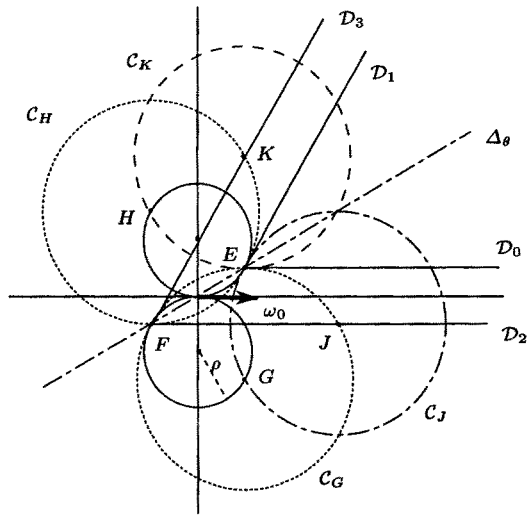


Fig. 19. Particular points, lines and circles

Then we have:

- the *lsl* domain is the internal angular sector defined by $\mathcal{D}_0$ and $\mathcal{D}_1$,
- the *rsr* domain is the external angular sector defined by $\mathcal{D}_2$ and $\mathcal{D}_3$,
- the *rsl* domain is the exterior of circle $\mathcal{C}_G$,
- the *lrl* domain is the union of the intersections between the pairs of discs $\mathcal{C}_G$ and $\mathcal{C}_H$, $\mathcal{C}_G$ and $\mathcal{C}_J$, $\mathcal{C}_H$ and $\mathcal{C}_K$,
- the *rlr* domain is the union of discs $\mathcal{C}_G$ and $\mathcal{C}_H$.

Notice that these domains intersect each other and do not partition P_θ. Each domain is defined upon two parameters. By fixing one parameter as the other one varies, we trace iso-parametric curves creating a foliation of each domain. From this construction it appears that a unique path of each type starts from each point located in the interior of the domain.

5.3 Construction of the partition

At this stage, we have to determine which path type is actually optimal in each region of P_θ covered by more than one domain. We are sure of the optimality of the whole *lsl* domain for the *lsl* type, even if other domains intersect it. Indeed, this domain is optimal for RS, and Dubins' sufficient family is included in the Reeds and Shepp one (except for CCC paths, which are shortened by $C|C|C$ paths). Clearly, a path type with no cusp which is optimal for RS is a fortiori optimal for DU.

So, let us consider the other intersections. Due to the lack of the symmetry with respect to the Δ_θ-axis, we cannot use a geometric reasoning to compare isometric trajectories, as we did in section 4.3 for RS. Therefore, in each region where more than one path type occur (see fig. 19), we use the method developed at section 4.3 for RS to compute the boundaries of the subdomains in which each path type is optimal. This method is based on the computation of the set of points reachable by a path of each type having the same length and starting at the origin. We conclude with arguments of continuity based on the foliation of domains by iso-length curves.

We will only present here the final equations of these boundary curves, since the calculation are really tedious (see [11] for more details).

Intersection *rsr* / *rsl* The intersection of these two domains, is defined by the complementary in P_θ to the set made by the union of the disc $\mathcal{C}_G$ and the internal angular sector defined by the rays $\mathcal{D}_2$ and $\mathcal{D}_3$. Writing that the final point are identical, and that both curves have the same length, we get a system of three equations with four variables. Fixing one variable as a parameter, we obtain the parametric expression of a curve $\mathcal{I}_0$:

$$\mathcal{I}_0 \quad \begin{cases} x = \lambda \cos a + 2 \sin a + \sin\theta \\ y = -\lambda \sin a + 2 \cos a - \cos\theta - 1 \end{cases}$$

where $\lambda = \frac{\rho(a+\theta-\pi)^2+2(\cos(a+\theta)-1)}{\sin(a+\theta)-(a+\theta-\pi)}$, and a is the length of the first arc in the *rsl* path. This parameter varies within the interval $]\pi-\theta, \mu]$ for $\theta \in [0, \pi[$ where μ is defined as follows:

- for $\theta \leq \frac{\pi}{2} + \pi$, $\mu = \pi - \theta + \eta$ where η is the solution of the non-algebraic equation $\cos t = t$,
- for $\theta \geq \frac{\pi}{2} + \pi$, μ is the value of a obtained when $\mathcal{I}_0$ and $\mathcal{D}_2$ intersect. This value can be computed by equating the parametric system of both curves.

This curve divides the region of intersection into two sub-domains, and admits the line of orientation θ, passing through G, as asymptote. We define the symmetric curve $\mathcal{I}_1$ for the intersection between *rsr* and *lsr*.

Intersection *rsl* / *lsr* Let η defined as in section 5.3. For $\theta \leq \frac{\pi}{2} + \eta$ we deduce from the analysis of iso-distance curves of each type that *lsr* paths are always shorter than *rsl* paths in the infinite region delimited by $\mathcal{D}_1$, $\mathcal{D}_3$, and the arc (E, K) of circle $\mathcal{C}_H$. Symmetrically with respect to the $\Delta_\theta^\perp$- axis, in the infinite region delimited by $\mathcal{D}_0$, $\mathcal{D}_2$, and the arc (E, J) of circle $\mathcal{C}_G$ the paths *rsl* are shorter than the *lsr* ones.

For $\theta > \frac{\pi}{2} + \pi$ a new boundary curve $\mathcal{I}_6$ appears; it is the locus of points reachable from the origin by a path *rsl* and *lsr* having the same length. This curve is determined by equating the parametric system of both curves. The curve $\mathcal{I}_7$ is obtained by symmetry with respect to $\Delta_\theta^\perp$ (see fig. 24)

Intersection *rsr* / *rlr* This region of intersection is made up by the parts of the discs $\mathcal{C}_G$ and $\mathcal{C}_H$ lying inside the external angular sector defined by $\mathcal{D}_2$ and $\mathcal{D}_3$. We find geometrically that the set of points reachable from the origin by a path of each type *rsr* and *rlr* having the same length belongs to a circle called $\mathcal{I}_2$ of radius 4η and centered at F. Thus, this set is made of two arcs of the circle $\mathcal{I}_2$ respectively defined by the interval of polar angles: $[\max(\theta, \pi/2 - \eta), \min(\theta, \pi/2 + \eta)]$ and the symmetric interval w.r.t. $\theta/2$. This intersection only occurs if $\theta \leq \pi/2 + \eta$.

Intersection *rlr* / *lrl* Using the same reasoning as in the study of the first intersection, we deduce that inside the region determined by the union of the intersections of discs $\mathcal{C}_G$ and $\mathcal{C}_J$, and the intersection of discs $\mathcal{C}_H$ and $\mathcal{C}_K$, *rlr* paths are always shorter than *lrl* paths. However, for $\theta > \pi/2$, the region of intersection of discs $\mathcal{C}_G$ and $\mathcal{C}_H$ is divided into two subdomains by a curve called $\mathcal{I}_3$. Paths *rlr* are optimal in the first subdomain, whereas paths *lrl* are optimal in the other one.

After a change of variables, due to the rotation of angle $\theta/2$, we obtain the following parametric equations for $\mathcal{I}_3$:

$$\mathcal{I}_3 \begin{cases} X = \dfrac{\cos v + \cos(v + \theta)}{\sin \frac{\theta}{2}} \\ Y^2 = (4 \sin \frac{v}{2})^2 - (X - 2 \sin \frac{\theta}{2})^2 \end{cases}$$

where v is the length of the middle arc of the *lrl* path. See [11] for the detail relative to the determination of the interval in which v varies.

Intersection *rlr* / *rsl* This last intersection occurs in the region of the disc $\mathcal{C}_H$ located outside the disc $\mathcal{C}_G$ and inside the internal angular sector delimited by $\mathcal{D}_2$ and $\mathcal{D}_3$. Using the same method as before we determine a curve $\mathcal{I}_4$ delimiting two subdomains in which *rlr* and *rsl* are respectively optimal . The

curve $\mathcal{I}_5$ symmetric of $\mathcal{I}_4$ with respect to Δ_θ^+ determines the boundary between the subdomains of paths *rlr* and *lsr* in the symmetric region. The parametric equations of $\mathcal{I}_4$ are:

$$\mathcal{I}_4 \begin{cases} x = 2(\sigma \cos a + \sin a) + \sin\theta \\ y = 2(\sigma \sin a - \cos a) - \cos\theta - 1 \end{cases}$$

$$\begin{aligned} \text{with } a &= 2\pi - \arccos\alpha \\ \alpha &= \frac{\sigma^2 A + B\sqrt{4(1+\cos\sigma)^2 + 4(\sigma+\sin\sigma)^2 - \sigma^4}}{2(A^2+B^2)} \\ A &= \cos\theta(1+\cos\sigma) - \sin\theta(\sigma+\sin\sigma) \\ B &= \cos\theta(\sigma+\sin\sigma) + \sin\theta(1+\cos\sigma) \end{aligned}$$

where a is the length of the first arc and 2σ the length of the line segment in the *rsl* path. We can notice that, here again, equations are non-algebraic. This intersection only occurs for $\theta \geq \pi/2 - \eta$. See [11] for the determination of the range of σ.

5.4 Description of the partition

With the refinement provided by the previous section we finally obtain a partition of P_θ, for values of θ having a representative modulus 2π in $[0, \pi]$. Using the symmetry properties given by lemmas 20 and 21 we obtain the partition for any $\theta \in S^1$. The shape of domains varies continuously with respect to θ. In the sequel we describe four successive states of the partition according to four successive intervals of θ. We also describe the cross sections corresponding to two particular values: $\theta = 0$ and π:

- $\theta = 0$ (Figure 20)
  All the domains are represented but notice that, for three of them, not only one but two equivalent optimal paths are defined at each point. Notice also that, in fact, the *lrl* and *lsl* domains are not connected: the initial configuration O can be viewed as a point of the domain *lrl* (isolated in P_0), and the horizontal half-line ($x \geq 0, y = 0$) also belongs to the domain *lsl*.
- $\theta \in]0, \pi/2 - \eta]$ (Figure 21)
  For $\theta \neq 0$, a unique path type is defined in each domain, but some domains are not connected (*rlr*, *rsl* and *lsr*). For $\theta = \pi/2 - \eta$, the segment of $\mathcal{D}_2$ (*resp.* $\mathcal{D}_3$) and $\mathcal{I}_2$ intersect each other on $\mathcal{C}_G$ (*resp.* $\mathcal{C}_H$).
- $\theta \in]\pi/2 - \eta, \pi/2]$ (Figure 22)
  Here, the intersection curves $\mathcal{I}_4$ and $\mathcal{I}_5$ appear, and for $\theta = \pi/2$ the two crescents of the *rlr* domain are connected at one point on the $\Delta_{\pi/2}$ axis.
- $\theta \in]\pi/2, \pi/2 + \eta]$ (Figure 23)
  The intersection curve $\mathcal{I}_3$ has appeared between *rlr* and *lrl*. Everything varies continuously until $\mathcal{I}_2$ disappear when $\theta = \pi/2 + \eta$, since the segment

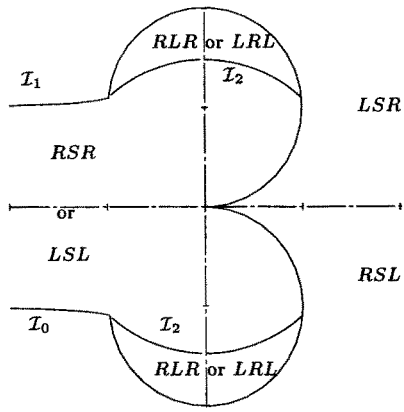


Fig. 20. Partition of P_0

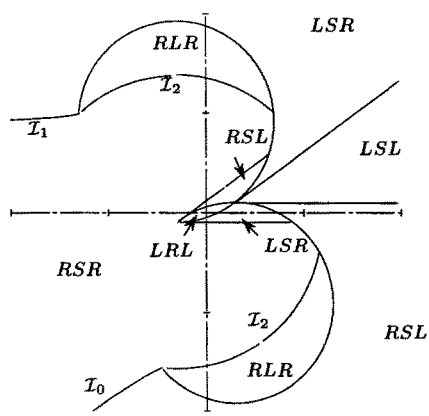


Fig. 21. Partition of $P_{\frac{\pi}{5}}$

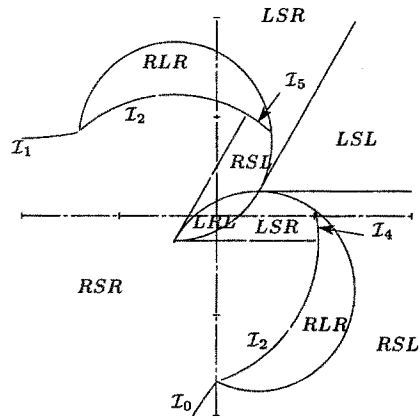


Fig. 22. Partition of $P_{\frac{\pi}{3}}$

of $\mathcal{D}_2$ (*resp.* $\mathcal{D}_3$), $\mathcal{I}_4$ (*resp.* $\mathcal{I}_5$), $\mathcal{I}_0$ (*resp.* $\mathcal{I}_1$) and the circle $\mathcal{C}_G$ (*resp.* $\mathcal{C}_H$) are concurrent.

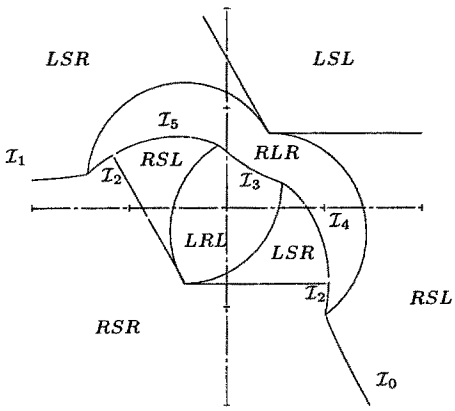


Fig. 23. Partition of $P_{\frac{2\pi}{3}}$

- $\theta \in]\pi/2 + \eta, \pi[$ (Figure 24)
  Domains are still varying continuously until $\mathcal{I}_4$ and $\mathcal{I}_5$ disappear, $\mathcal{I}_4$ and $\mathcal{I}_5$ become horizontal half-lines, and $\mathcal{I}_3$ becomes an horizontal segment of length 4.
- $\theta = \pi$ (Figure 25)

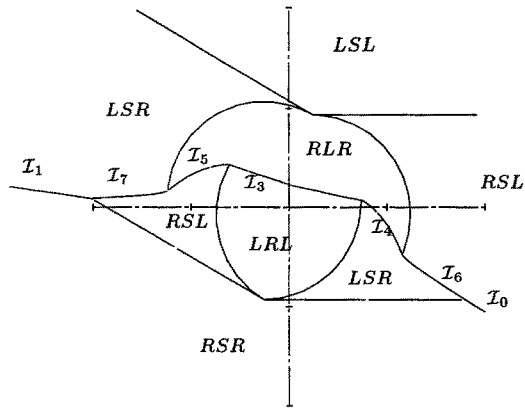


Fig. 24. Partition of $P_{\frac{5\pi}{6}}$

In this case the partition contains six types; the domains of paths *lsr* and *rsl* are still not connected.

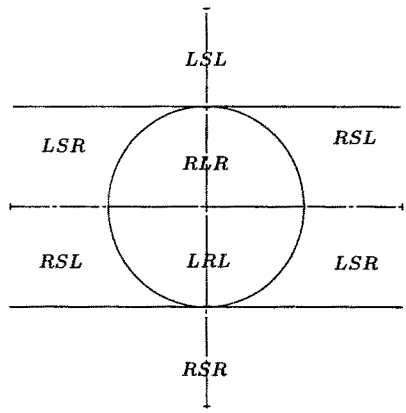


Fig. 25. Partition of P_π

Analysing this construction we can make the following remarks:

1. Optimal domains are not necessarily connected, unlike the Reeds and Shepp case. This is due to the fact that a configuration (x, y, θ) can sometimes be reached in different ways: either mostly turning left until the algebraic sum

of angles equals θ, or mostly turning right so that the algebraic sum equals $2\pi - \theta$. For the Reeds and Shepp case, these two solutions cannot be both optimal since the algebraic sum of angles has to be lower than π.

2. The shape of the shortest paths varies continuously when crossing the boundary of any domain, except the boundary arcs of discs $\mathcal{C}_G$ and $\mathcal{C}_H$, and the intersection curves $\mathcal{I}_i$.
3. The shortest path's length is a continuous function of (x, y, θ) everywhere, *except* on the boundary arcs of discs $\mathcal{C}_G$ and $\mathcal{C}_H$. This discontinuity (in shape and length) is due to the fact that inside the circle $\mathcal{C}_G$ (*resp.* $\mathcal{C}_H$) the *rsl* (*resp. lsr*) path does not exist. Figure 26 represents the iso-distance curves in the plane P_θ for $\theta = 1$ *rad.* The two thicker arcs in the center of the picture represent the locus of points where the length function is discontinuous.
4. For $\theta = 0$, there exist two regions where two equivalent optimal solution are defined. Therefore, to define a synthesis of optimal paths (uniqueness of the solution), it suffice to choose arbitrarily a constant values for the control in each region where several optimal strategies are available.

Remark 11. *Due to the lack of continuity of the length function, this synthesis of optimal paths does not verify Boltianskii's regularity conditions (condition F of definition 9 fails). This illustrates the fact that the very strong hypotheses defining Boltianskii's regular synthesis restrict the application area to a very small class of problems. This example raises up the interest of searching for sufficient conditions, weaker than Bolitanskii's ones, that still guarantee the optimality of marked trajectories.*

5.5 Related works

Using also the frame of geometric control, R. Felipe Monroy Pérez has studied Dubins' problem in the case of Euclidean and non-Euclidean geometries [28].

In the Euclidean case (classical problem of Dubins) he provided a new proof of the non optimality of the concatenation of four arcs of circle. He proved that in two dimensional simply connected manifold with constant sectional curvature, trajectories of minimal length necessarily follow Dubin's pattern (CLC and CCC) where L denotes a piece of a geodesic and C an arc of curve with constant curvature. The study was done by means of optimal control on Lie groups.

For the three dimensional case, he exhibited an explicit expression of the torsion of optimal arcs. In particular, he determined a parametric equation of curves satisfying optimality conditions in $\mathbf{R}^3$, providing a representation of potential solutions for Dubins' problem in $\mathbf{R}^3$.

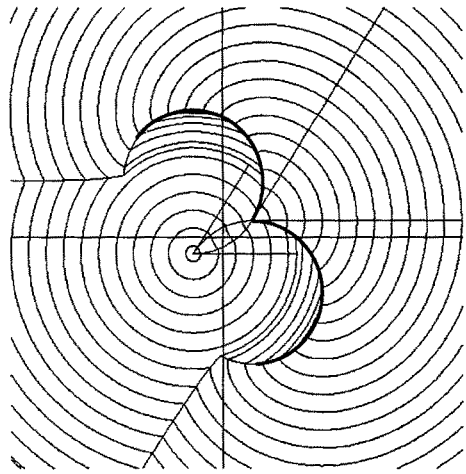


Fig. 26. Iso-distance curves in P_1 and discontinuity of the length function.

Dubins' problem in $\mathbf{R}^3$ has been also studied by H. J. Sussmann in [37]. By applying PMP on manifolds he proved that every minimizer is either an helicoidal arc or a path of the form CSC or CCC.

6 Dubins model with inertial control law

From the previous section we know that optimal solutions of Dubins' problem are sequences of line segments and arcs of circle of minimal radius. Therefore, there exist curvature discontinuities between two successive pieces, line-arc or arc-arc (with opposite direction of rotation) and to follow (exactly) such a trajectory a real robot would be constrained to stop at the end of each piece. In order to avoid this problem, Boissonnat, Cerezo and Leblond [3] have proposed a generalization of Dubins' problem by suggesting to control the angular acceleration of the car instead of its angular velocity. This section presents the analysis of the shortest paths problem for this model.

Using the same notation as for Dubins' problem, let $M(x, y)$ be the coordinates of the robot's reference point with respect to a fixed orthonormal frame, and θ its orientation with respect to the $x-$axis. We use $\kappa(t)$ to represent the signed curvature of the path at each time ($\kappa(t) > 0$, meaning that the car is turning left).

In the plane of the robot's motion we consider a class $\mathcal{C}$ of C^2 paths joining two given configurations $X_0 = (M_0, \theta_0, \kappa_0)$ and $X_f = (M_f, \theta_f, \kappa_f)$.

Definition 11. *A path belongs to class C if it satisfies the following two properties:*

1. Regularity: *the path is a C^2 concatenation of an at most countable number of open C^3 arcs of finite length, and the set of endpoints of these arcs, also called the* switching points, *admits at most a finite number of accumulation points.*
2. Constraint: *along the path, the absolute value of the derivative of the curvature , with respect to the arc length, is upper bounded by a given constant $B > 0$, at every point where it is defined.*

With these notations and the above definition, the motion of the oriented point $M(t) = (x(t), y(t), \theta(t), \kappa(t))$ along paths of class C in $\mathbf{R}^2 \times S^1 \times \mathbf{R}$ is well-defined and continuous.

In the sequel we consider that the robot moves at constant speed 1, so that time and arc length coincide.

A path in class C between any two configurations $X_0 = (x_0, y_0, \theta_0, \kappa_0)$ and $X_f = (x_f, y_f, \theta_f, \kappa_f)$, if it exists, is entirely determined by the function $v(t) = \dot{\kappa}(t)$, defined and continuous everywhere, except at the switching points, by the following differential system:

$$\dot{X}(t) = \begin{cases} \dot{x}(t) = \cos\theta(t) \\ \dot{y}(t) = \sin\theta(t) \\ \dot{\theta}(t) = \kappa(t) \\ \dot{\kappa}(t) = v(t) \end{cases} \tag{30}$$

If we add the boundary conditions $X(0) = X_0$, $X(f) = X_f$, and the constraint:

$$\forall t \in [0, T], \quad |v(t)| \leq B, \tag{31}$$

and if we search for a path of minimum length in class C, we have turned the geometric problem into a classical question of optimal control theory where the functional:

$$J(v) = T = \int_0^T dt \tag{32}$$

is to be minimized among the set of control functions v satisfying (31).

6.1 Existence of an optimal solution

System (30) may be written as:

$$\dot{X} = F(X, v) = f(X) + v\, g(X)\,,$$

where the analytic vector fields f and g are given by:

$$f(X) = \begin{pmatrix} \cos\theta \\ \sin\theta \\ \kappa \\ 0 \end{pmatrix}, \; g(X) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Complete controllability of the system We first observe that the Lie algebra $\mathcal{L}(f,g)$ generated by f and g is, at each point, of dimension 4. Indeed, $\forall X \in \mathbf{R}^4$,

$$h(X) = [g,f](X) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \; i(X) = [h,f](X) = \begin{pmatrix} -\sin\theta \\ \cos\theta \\ 0 \\ 0 \end{pmatrix},$$

and

$$\det \begin{bmatrix} \cos\theta & 0 & 0 & -\sin\theta \\ \sin\theta & 0 & 0 & \cos\theta \\ \kappa & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = -1.$$

Moreover, the solutions of the associated autonomous system $\dot{X} = f(X)$ are circles (of radius $1/\kappa$), thus periodic. Hence, Bonnard's theorem [27, thm.III.4] applies, to establish the complete controllability of (30) under the constraint (31). This means that any X_0 and X_f can always be joined by a path satisfying (30) and (31).

Existence of an optimal control The existence of an optimal control for the problem (30), (31), (32), with given X_0 and X_f, is ensured by Fillipov's existence theorem (see [13, 5.1.ii] for example). Indeed, the hypotheses of the theorem are satisfied. The dynamic $F(X,v)$ and the cost $J(v)$ are smooth enough, the set $[-B,+B]$ of control is convex, and the initial and final configurations X_0 and X_f are fixed. Finally, one can easily check the existence of a constant C such that ${}^tX\,F(X,v) \leq C\,(|X|^2+1)$ for all $t \in [0,T]$, $X \in \mathbf{R}^2 \times S^1 \times \mathbf{R}$, $v \in [-B,+B]$.

Fillipov's theorem then asserts the existence of some $T^* > 0$ and of an optimal control $v^*(t)$ which is a measurable (thus locally integrable) function which satisfies (31) on $[0,T^*]$. The solution of (30) for $v = v^*$ is a path from X_0 to X_f which minimizes cost (32) under constraint (31).

6.2 Necessary conditions for a solution to be optimal

Pontryagin's Maximum Principle We are going to apply Pontryagin's Maximum Principle in order to obtain necessary conditions for a solution to be optimal (i.e. a measurable control v and a trajectory X) minimizing cost (32).

Let us denote by Ψ, ${}^t\Psi = (\psi_1, \psi_2, \psi_3, \psi_4)$, the adjoint state associated to X. For this minimum time problem, the Hamiltonian H is defined for every $t \in [0, T]$ by

$$H(\Psi(t), X(t), v(t)) = < \Psi(t), F(X(t), v(t)) >$$

This yields in the case of system (30):

$$H(\Psi(t), X(t), v(t)) = \psi_1(t)\cos\theta(t) + \psi_2(t)\sin\theta(t) + \psi_3(t)\kappa(t) + \psi_4(t)v(t). \tag{33}$$

The adjoint state Ψ is defined on $[0, T]$ as a solution to the adjoint system $\dot{\Psi} = -\frac{\partial H}{\partial X}$, which is here:

$$\dot{\Psi}(t) = \begin{cases} \dot{\psi}_1(t) = 0 \;\Rightarrow \psi_1(t) = \psi_1 \\ \dot{\psi}_2(t) = 0 \;\Rightarrow \psi_2(t) = \psi_2 \\ \dot{\psi}_3(t) = \psi_1(t)\,\sin\theta(t) - \psi_2(t)\,\cos\theta(t) = \psi_1\dot{y}(t) - \psi_2\dot{x}(t) \\ \dot{\psi}_4(t) = -\psi_3(t)\,. \end{cases} \tag{34}$$

Therefore, as ψ_1 and ψ_2 are constant on $[0, T]$ there exists $\lambda \geq 0$ and $\phi \in [0, 2\pi[$ such that, $\forall t \in [0, T]$:

$$\begin{cases} \psi_1(t) \equiv \psi_1 = \lambda\cos\phi \\ \psi_2(t) \equiv \psi_2 = \lambda\sin\phi \\ \dot{\psi}_3(t) = \lambda\sin(\theta(t) - \phi) \\ \dot{\psi}_4(t) = -\psi_3(t)\,. \end{cases} \tag{35}$$

The Hamiltonian (33) can now be written as:

$$H(\Psi, X, v) = \lambda\cos(\theta - \phi) + \psi_3\kappa + \psi_4 v. \tag{36}$$

Now, according to theorem 3, a necessary condition for $X(T)$ to be an extremal trajectory for the minimum-time problem is that $\Psi(t)$ define a nonzero absolutely continuous function such that $\forall t \in [0, T]$:

$$H(\Psi(t), X(t), v(t)) = \max_{u \in [-B, +B]} H(\Psi(t), X(t), u(t)) = -\psi_0. \tag{37}$$

where $\psi_0 \leq 0$ is a constant.

Characterization of extremal arcs From equation (37) we deduce that:

$$\psi_4(t)\, v(t) \geq 0 \text{ for almost every } t \in [0,T]\,. \tag{38}$$

As X belongs to class $\mathcal{C}$, on each open C^3 portion of the trajectory, $v(t) = \pm B$ with the sign of ψ_4 if $\psi_4(t) \neq 0$ or, otherwise, that $\frac{\partial H}{\partial v} = \psi_4(t) = 0$. If $\psi_4(t) \equiv 0$ over some interval $[t_1, t_2] \subset [0,T]$, (35) implies that $\psi_3(t) \equiv 0$ and $\dot{\psi}_3(t) \equiv 0$. As θ is continuous and $\lambda \neq 0$ (otherwise $\psi_1 = \psi_2 = \psi_3 = 0$ and therefore $\psi_0 = 0$ which is not possible), it follows that $\theta(t) \equiv \phi \pmod{\pi}$. Of course then, $\kappa \equiv v \equiv 0$ on $[t_1, t_2]$. Hence, on each open C^3 portion of the path, $v(t) \in \{-B, +B, 0\}$, and since v has to be continuous on such a portion, it is of one of the three kinds:

1. Cl^+: $v(t) \equiv B$, $\psi_4(t) > 0$
2. Cl^-: $v(t) \equiv -B$, $\psi_4(t) < 0$
3. S: $v(t) \equiv 0$, $\psi_4(t) \equiv 0$

Arcs $\mathrm{Cl}^{\pm}$ are finite portions of *clothoids*. A clothoid[6] (see figure (27)), also known as a "Cornu spiral", is a curve along which the curvature κ depends linearly on the arc length (here equal to t) and varies continuously from $-\infty$ to $+\infty$. Hence, all clothoids Cl^+ (where $v(t) = B$) are translated and rotated copies of a unique clothoid Γ while all clothoids Cl^- (where $v(t) = -B$) are translated, rotated and reflected copies of Γ. Clothoids Cl^+ will be called *direct* clothoids and clothoids Cl^- will be called *indirect* clothoids. The canonical clothoid Γ is chosen as the one defined by the following equations:

$$x(t) = \int_0^t \cos(\frac{B}{2}\tau^2)d\tau$$
$$y(t) = \int_0^t \sin(\frac{B}{2}\tau^2)d\tau\,.$$

Arcs S are line segments, all with the same orientation $\phi \pmod{\pi}$.
From the above discussion, we have:

Proposition 1. Any extremal path in class $\mathcal{C}$ is the C^2 concatenation of line segments (with the same orientation) and of arcs of clothoids (with $\dot{\kappa} = \pm B$), all of finite length. The control function v is constant on each piece: $v = B$ on a direct clothoid Cl^+, $-B$ on an indirect one Cl^-, and 0 on a line segment S.

In the sequel, we denote by "Cl" an arc of clothoid, by "S" an open line segment, and by "." a switching point. "Cl_μ" will further specify, when necessary, the length μ of the arc.

[6] More details about clothoiods are given in the next section

In order to characterize the extremal paths, and, among them, the shortest ones, we consider the following problem: *how are these arcs Cl and S arranged together along an extremal trajectory of class C ?*
We provide in the next section a partial answer to this question.

Concatenation of arcs

Lemma 25. $\psi_4 = 0$ at any switching point (Cl.Cl, Cl.S or S.Cl).

Proof: That $\psi_4 = 0$ at a switching point Cl.S or S.Cl follows from the fact that $\psi_4 \equiv 0$ on S and that ψ_4 is continuous. At a switching point Cl.Cl, the sign of v changes and, by (38), also the sign of ψ_4. □

Lemma 26. If $\lambda = 0$, the extremal path consists of one or two arcs and is of type Cl or Cl.Cl.

Proof: If $\lambda = 0$, ψ_3 is constant on $[0, T]$ by (35). If $\psi_3 = 0$, ψ_4 is constant on $[0, T]$ by (35). Moreover, ψ_4 cannot be identically 0, since, otherwise, $(\Psi, \psi_0) \equiv (0, 0)$, which contradicts the necessary conditions of PMP. Hence, it follows from Lemma 25 that the extremal path cannot contain a line segment nor a switching point and thus reduces to a single arc Cl.

If $\psi_3 \neq 0$, $\psi_4(t)$ is a linear function of t by (35) and then vanishes at most at one isolated point. Hence the extremal path is of type Cl or Cl.Cl, by Lemma 25. □

Note that such paths are not *generic*: from any given initial configuration X_0 in $\mathbf{R}^2 \times S^1 \times \mathbf{R}$, the set of final configurations $\{X_f\}$ one can reach through such paths is only 1 or 2-dimensional.

Lemma 27. If an extremal path contains a line segment S, $\lambda = -\psi_0 > 0$.

Proof: along a line segment $\psi_4 \equiv \psi_3 \equiv 0$ and $\theta \equiv \phi \pmod{\pi}$. Hence, $H \equiv \varepsilon\lambda = -\psi_0 \geq 0$, with $\varepsilon = \pm 1$. As $\psi_0 \leq 0$ and $\lambda > 0$ (from Lemma 26), we must have $\varepsilon = +1$ and $\lambda = -\psi_0$. □

From the proof of lemma 27, $\varepsilon = \cos(\theta - \phi) = +1$ on S, and we have:

Corollary 1. Along a line segment S, $\theta \equiv \phi \pmod{2\pi}$.

Lemma 28. $\psi_3 - \psi_1 y + \psi_2 x$ is constant along any extremal path. If $\lambda > 0$, for any given $c \in \mathbf{R}$, all the points of an extremal path where $\psi_3 = c$ lie on the same straight line D_c, of direction $\phi \pmod{\pi}$.

Proof: $\dot{\psi}_3 = \psi_1 \dot{y} - \psi_2 \dot{x}$ from (34), and ψ_1 and ψ_2 are constant. Thus there exists a constant c_0 such that $\psi_1 y - \psi_2 x = \psi_3 + c_0$, which proves the first part of the lemma. If $\lambda \neq 0$, ψ_1 and ψ_2 cannot be both equal to 0 and $\psi_1 y - \psi_2 x = c + c_0$ is the equation of a line of direction $\theta = \phi \pmod{\pi}$. □

As a consequence, we have:

Corollary 2. Any line segment S of an extremal path is contained in D_0 and is run with $\theta \equiv \phi \pmod{2\pi}$.

Proof: since $\psi_3 \equiv 0$ on S, it follows from Lemma 28 that S is contained in the line D_0 of direction ϕ. By Corollary 1, $\theta \equiv \phi \pmod{2\pi}$. □

Lemma 29. If $\lambda > 0$, each open arc of clothoid Cl_μ with $\mu > 0$ of an extremal path, except possibly the initial and the final ones, intersects D_0 at least once.

Proof: let Cl_μ be an arc of length μ of an extremal path which is not the initial nor the final arc. Both endpoints of such an intermediate arc are switching points. Let $]t_1, t_2[$ denote the time interval during which this intermediate arc Cl_μ is run. By Lemma 25, $\psi_4(t_1) = \psi_4(t_2) = 0$. As $t_2 - t_1 = \mu > 0$, there exists at least one $t \in]t_1, t_2[$, say t_3, such that $\dot{\psi}_4(t_3) = 0$ and thus, from (35), $\psi_3(t_3) = 0$. Finally, it follows from Lemma 28 that $M(t_3)$ belongs to D_0. □

Observe that the hypothesis $\lambda > 0$ along an extremal path is true as soon as it contains either a line segment (Lemma 27) or more than only two arcs of clothoid (Lemma 26).

Lemma 30. An extremal path contains no portion of type $\mathrm{S.Cl}_\mu\mathrm{.Cl}$ or of symmetric type $\mathrm{Cl.Cl}_\mu\mathrm{.S}$ with $\mu > 0$.

Proof: assume that there exists such a portion $\mathrm{S.Cl}_\mu\mathrm{.Cl}$ and let $]t_1, t_2[$ denote the time interval during which Cl_μ is run, with $t_2 - t_1 = \mu > 0$. From Lemma 2, $\mathrm{S} \subset D_0$, and since the variables (x, y, θ, κ) are continuous on $[t_1, t_2]$, Cl_μ is tangent to D_0 at $M(t_1)$ and $\kappa(t_1) = 0$. Hence, $M(t_1)$ is the inflection point of the clothoid supporting Cl_μ and D_0 is the tangent to the clothoid at $M(t_1)$. This implies that $\mathrm{Cl}_\mu \setminus \{M(t_1)\}$ is entirely contained in an open half-plane delimited by D_0, see figure (27), which contradicts Lemma 29. □

The last lemma is in fact superseded by the following one, due to H.J. Sussmann.

Lemma 31. An extremal path contains no portion of type $\mathrm{S.Cl}_\mu$ (or $\mathrm{Cl}_\mu\mathrm{.S}$) with $\mu > 0$.

Proof: assume that there is a portion of type $\mathrm{S.Cl}_\mu$, with $\mu > 0$, in an extremal trajectory and let t_1 be the switching time between S and Cl_μ. From (35) and (36) we obtain the following expressions of the four first derivatives of ψ_4 (valid on S as well as on Cl_μ):

$$\begin{cases} \ddot{\psi}_4 = -\lambda \sin(\theta - \phi) \\ \dddot{\psi}_4 = -\lambda \kappa \cos(\theta - \phi) \\ \ddddot{\psi}_4 = \lambda \kappa^2 \sin(\theta - \phi) + (\psi_3 \kappa + \psi_4 v + \psi_0) v \,. \end{cases}$$

Hence, the adjoint variable ψ_4 is of class C^3 in the neighbourhood of t_1. Moreover, on S, $\psi_4 = \dot{\psi}_4 = 0$, $\theta \equiv \phi \pmod{2\pi}$, $\psi_3 = 0$, $\kappa = 0$. From the above equations, we also have $\ddot{\psi}_4 = \dddot{\psi}_4 = 0$ on S, and, by continuity, at t_1. Moreover, $\ddddot{\psi}_4\,(t_1) = \psi_0\, v$. Thus, there exists an ε, $0 < \varepsilon \leq \mu$, such that for $t \in [t_1, t_1 + \varepsilon[$ we have:

$$\psi_4(t) = \psi_0\, v(t)\, \frac{(t - t_1)^4}{4!} + o((t - t_1)^5)\,.$$

Now, from Lemma 27, $\psi_0 < 0$, so that ψ_4 and v have opposite signs on $[t_1, t_1 + \varepsilon[$ which contradicts (38). □

A consequence of Lemmas 30 and 31 is the following proposition:

Proposition 2. If an extremal path of class $\mathcal{C}$ contains but is not reduced to a line segment, then it contains an infinite number of concatenated clothoid arcs which accumulate towards each endpoint of the segment which is a switching point.

Proposition 2 together with the fact that a clothoid Cl is contained in a ball of bounded diameter D_{Cl} (depending on the parameter B) implies the following:

Proposition 3. The number n of C^3 pieces contained in a generic extremal path cannot be uniformly bounded from above (with respect to X_0, X_f). However, if $d(M_0, M_f)$ denotes the Euclidean distance in the plane between M_0 and M_f, we have that:

$$n \geq \frac{d(M_0, M_f)}{D_{Cl}}\,.$$

Proof: either the shortest path contains (and is generically not reduced to) a line segment, and Proposition 2 implies that there are infinitely many arcs of clothoid, or it is made only with arcs of clothoid, the number of which clearly depends on (and increases with) the distance between X_0 and X_f. The bound from below is obvious. □

6.3 Conclusion

Note that it is not clear whether or not extremal trajectories described in Proposition 2, and, among them, the optimal ones, belongs to class C: indeed, the set of switching points on an optimal trajectory (points where the control v is undefined) might even be uncountable. Moreover, we don't know yet if the statement of this proposition remains true without the assumption that the path contains a line segment.

However, Propositions 2 and 3 already indicate that the optimal control associated to problem (30), (31), (32) has a complex behavior. Contrarily to what occurs for Dubins or Reeds and Shepp problems, for which every optimal trajectory contains at most a prescribed (finite) number of line segment and arcs of circles, the number of switching points is unbounded here and might be infinite.

6.4 Related works

Lemma 31 is due to H.J. Sussmann who provided a complete study of this problem described as "Markov-Dubins problem with angular acceleration control" [38]. In this paper, the author uses results by Zelikin and Borisov [39] to show that there exist extremals involving infinite chattering.

7 Time-optimal trajectories for Hilare-like mobile robots

The last model we consider is the model of Hilare the robot of LAAS-CNRS whose locomotion system consists of two parallel driven wheels and four slave castors.

Let (x, y) be the coordinates of the reference point located between the driven wheels, and θ the robot's orientation with respect to the $x-$axis. v_r and v_l denote respectively the velocities of the contact point of the right and left driven wheel with the floor. These virtual point velocities are considered as two state variables while their acceleration a_r and a_l constitute the two system inputs. Therefore, a configuration of the robot is a 5-uple (x, y, θ, v_r, v_l). Using d to denote the distance between the driven wheels we get the following dynamic representation:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_r \\ \dot{v}_l \end{pmatrix} = \begin{pmatrix} \frac{v_r+v_l}{2}\cos\theta \\ \frac{v_r+v_l}{2}\sin\theta \\ \frac{v_r-v_l}{d} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} a_r + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} a_l \tag{39}$$

Each wheel is driven by an independent motor and the power limitation is expressed by the constraint: $a_r, a_l \in [-a, a], a > 0$. For this model there is no curvature constraint and the robot can turn about its reference point.

We consider the problem of characterizing minimum-time trajectories linking any pair of configurations where the robot is at rest i.e verifying $v_r = v_l = 0$.

This problem has been initially studied by Jacobs *et al* [25]. After having shown that the system is controllable, the authors have proven that minimum-time trajectories are necessarily made up with bang-bang pieces. To illustrate

the reasoning of their proof let us suppose that the first control a_r is singular while the second control a_l is bang-bang.

For this minimum-time problem, denoting by $\psi = (\psi_1, \psi_2, \psi_3, \psi_4, \psi_5)^T$ the adjoint vector, the Hamiltonian corresponding to system (39) is:

$$H = \psi_1 \frac{v_r + v_l}{2} \cos\theta + \psi_2 \frac{v_r + v_l}{2} \sin\theta + \psi_3 \frac{v_r - v_l}{2} + \psi_4 \, a_r + \psi_5 \, a_l$$

As we suppose a_r to be singular, the corresponding switching function ψ_4 vanishes over a nonzero interval of time. From the adjoint equation we get:

$$\dot{\psi}_4 = -\frac{\partial H}{\partial v_r} = \frac{\psi_1}{2} \cos\theta + \frac{\psi_2}{2} \sin\theta + \frac{\psi_3}{d} = 0$$

and therefore,

$$\psi_3 = -\frac{d}{2}(\psi_1 \cos\theta + \psi_2 \sin\theta)$$

Taking the derivative of ψ_3 and replacing $\dot{\theta}$ by its expression given by (39) we get:

$$\dot{\psi}_3 = (\psi_1 \sin\theta - \psi_2 \cos\theta)(\frac{v_r - v_l}{2}) \tag{40}$$

The expression of $\dot{\psi}_3$ can also be deduced directly from the adjoint equation:

$$\dot{\psi}_3 = -\frac{\partial H}{\partial \theta} = (\psi_1 \sin\theta - \psi_2 \cos\theta)(\frac{v_r + v_l}{2}) \tag{41}$$

Equating (40) and (41) we deduce that

1. either $v_2 \equiv 0$
2. either $\psi_1 \sin\theta - \psi_2 \cos\theta = 0$

As a_l is supposed to be bang-bang the first case leads to a contradiction. On the other hand, as $\frac{\partial H}{\partial x} = \frac{\partial H}{\partial y} = 0$, we deduce from the adjoint equation that ψ_1 and ψ_2 are constant. Thus, in the second case, the car is moving on a straight line, but a necessary condition for such a motion to be time-optimal is that the acceleration of wheels be both maximal or both minimal, and therefore correspond to bang-bang control.

Using the same reasoning in the case that a_l is singular and a_r regular, or in the case that both control are singular, one can prove that extremal controls are necessarily bang-bang.

Therefore, optimal trajectories are obtained for $|a_r| = |a_l| = a$; these extremal curves are of two types.

- $a_r = -a_l = \pm a$
  In this case the robot's linear acceleration is null: $\dot{v}(t) = \frac{1}{2}(\dot{v}_d + \dot{v}_g) = 0$. $v(t)$ is constant equal to v_0, therefore the curvilinear abscissa $s(t) = v_0 t$. $\dot{v}_r(t) = \pm a$ while $\dot{v}_l(t) = \mp a$. Integrating we get: $v_r(t) = \pm at + v_{r0}$, $v_l(t) = \mp at + v_{l0}$ and $\omega(t) = \dot{\theta}(t) = \pm\frac{2a}{d}t + \omega_0$. The curvature κ is then:

$$\kappa(t) = \frac{\pm\frac{2a}{d}t + \omega_0}{v_0} = \pm k_c s(t) + \frac{\omega_0}{v_0} \tag{42}$$

where $k_c = \frac{2a}{dv_0^2}$. In the (x, y)−plane, the curve is a clothoid with characteristic constant k_c. When $x_0 = y_0 = \omega_0 = \theta_0$ the curve is expressed by the following parametric expression in terms of Fresnel sine and cosine.

$$\begin{cases} x(t) = sign(v_0)\sqrt{\frac{\pi}{k_c}}\int_0^{\sqrt{\frac{2a}{d\pi}}t} \cos(\frac{\pi}{2}\tau^2)d\tau \\ y(t) = sign(v_0 a_r)\sqrt{\frac{\pi}{k_c}}\int_0^{\sqrt{\frac{2a}{d\pi}}t} \sin(\frac{\pi}{2}\tau^2)d\tau \end{cases} \tag{43}$$

Figure (27) shows a clothoid obtained for $a_r = -a_l = a$. The part located above the x−axis describes the robot's motion for $v_0 > 0$, while the part located under the $x-$ axis corresponds to $v_0 < 0$. A curve symmetric with respect to the x−axis is obtained for $a_r < 0$.

Remark 12. *When $v_0 = 0$ the curve is reduced to a pure rotation about the origin.*

- $a_r = a_l = \pm a$
  In this case the angular velocity is null: $\dot{\omega}(t) = \frac{1}{d}(\dot{v}_r(t) - \dot{v}_l(t)) = 0$. Therefore $\omega(t) = \omega_0$, and $\theta(t) = \omega_0 t + \theta_0$. The linear acceleration is $\dot{v}(t) = sign(a_r)a$, thus $v(t) = sign(a_r)at + v_0$. The curvature radius $\rho(t)$ is given by:

$$\rho(t) = \frac{v(t)}{\omega(t)} = sign(a_r)k_a(\theta(t) - \theta_0) + \frac{v_0}{\omega_0} \tag{44}$$

where $k_a = \frac{a}{\omega_0^2}$. In the (x, y)−plane the curve is an involute of a circle[7] whose characteristic constant is k_a. When $x_0 = y_0 = v_0 = \theta_0$ the curve is expressed by the following parametric expression:

$$\begin{cases} x(t) = sign(a_r)k_a(\cos(\omega_0 t) + \omega_0 t\sin(\omega_0 t) - 1) \\ y(t) = sign(a_r)k_a(\sin(\omega_0 t) - \omega_0 t\cos(\omega_0 t)) \end{cases} \tag{45}$$

[7] The involute of a circle is the curve described by the end of a thread as it is unwound from a stationary spool.

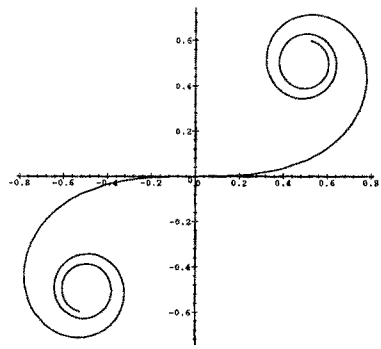


Fig. 27. clothoid obtained for $a_r = -a_l = a$

Figure (28) represents an involute of a circle obtained for $a_r = a_l = a$. The robot turns in the counterclockwise direction when $\omega_0 > 0$ and in the clockwise direction when $\omega < 0$. For $a_r < 0$ the resulting curve is symmetric with respect to the origin.

Remark 13. *When $\omega_0 = 0$, the curve is a line.*

This description achieves the local characterization of extremal curves. Optimal trajectories are made up with pieces of clothoids and involute of circles. The question is now to determine how many control switches occur along an optimal trajectory and how to determine the switching times. This difficult problem has motivated several research works.

A first work by Reister and Pin [30] was based on the conjecture that optimal paths contain at most four control switches. Using an interesting time parameterization they presented a numerical study of bang-bang trajectories containing only five elementary pieces. By computing the set of accessible configurations in fixed time they tried to state that trajectories containing more than five pieces are not optimal. Unfortunately this numerical analysis could not provide a mathematical proof to bound the number of control switches.

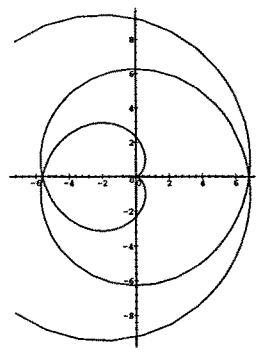


Fig. 28. Involute of circle obtained for $a_r = a_l = a$

More recently, the work by Renaud and Fourquet [32] has invalidated the conjecture by Reister and Pin, showing that certain configurations of the space could not be reached by extremal trajectories containing only five elementary pieces. Furthermore, they pointed out the existence of extremal solutions allowing to reach these configurations and containing more than four switches.

To our knowledge this work constitutes the last contribution to the problem. Therefore, to date, there does not exist any result allowing to bound the number of control switches along an optimal trajectory. It is then not possible at this stage to try to characterize a sufficient family as we did at section (4). In fact, the very first question we need to answer is to determine whether the number of switches is finite or not.

In spite of solving the minimum-time problem the local description of extremal curves can be used to deduce interesting geometric properties for path planning.

- Equation (42) show that clothoid allow to link smoothly curves with zero curvature (lines) and curves with nonzero curvature (arcs of circle).
- Equation (44) show that involutes of circle can link smoothly curves with infinite curvature (turn about) and curves with nonzero curvature. In par-

ticular, following this curve the robot can make a cusps while keeping a nonzero angular velocity.

This result has been used by Fleury *et al* [17] to design primitives for smoothing mobile robots' trajectories. In this work several sub-optimal strategies are proposed to smooth broken lines trajectories in a cluttered environment.

8 Conclusions

The study of these four problems corresponding to different models of wheeled robots illustrates the strengths and weaknesses of the use of optimal control for path planning.

By constructing a shortest paths synthesis for the models of Reeds and shepp and the model of Dubins, we have definitely solved the path planning problem for a car-like robot moving in a plane free of obstacles. Obviously, as the vehicle is supposed to move at a constant speed along arcs of circle and line segments this result does not constitute a real feedback control for the robot. However, it constitutes a canonical way to determine a path, for linking any two configurations, upon which path following techniques can be developed.

Furthermore, from this construction, it has been possible to determine a distance function providing a topological analysis of the path planning problem. In particular, for the Reeds and Shepp problem, we have proven that the distance induced by the shortest path is Lipschitz equivalent to a sub-Riemannian metric. Such a metric constitutes a very useful tool to compute the distance between the robot and its environment.

However, whereas optimal control may provide a very complete result for a small number of systems, the characterization of optimal path is in general incomplete. This is illustrated by the last two problems. In such cases, the local characterization of extremals can be used to determine suboptimal strategies for planning.

Beyond solving the path planing problem, this study has permitted to get very interesting results.

First, we have shown the existence of symmetry properties common to the different models of wheeled robots. On this basis, by constructing the set of reachable configuration for the model of Reeds and Shepp and for the model of Dubins, we have shown the existence of several propagating wave fronts intersecting each other. From this, we have proven the insufficiency of the local information provided by PMP and the need to be compare the cost of trajectories corresponding to different wave fronts, by means of global arguments. Using this reasoning we have completely solved the problem of Reeds and Shepp as well as the problem of Dubins.

On the other hand, by showing that the synthesis constructed for the Reeds and Shepp problem verifies the required regularity conditions we have found another proof to confirm this result *a posteriori* by applying Boltianskii's sufficient optimality conditions. Though this theorem allows to prove very strong results in a very simple way, we have shown the narrowness of its application area by considering the neighbouring example of Dubins for which the regularity conditions no longer apply because of the discontinuity of path length.

The last two examples illustrate the difficulty very often encountered in studying of optimal control problems. First, the adjoint equations are seldom integrable making only possible the local characterization of optimal paths. The search for switching times is then a very difficult problem. Furthermore, as we have seen in studying the problem of Dubins with inertial control, it is possible to face Fuller-like phenomenon though the solution could seem to be a priori intuitively simple.

References

1. L.D Berkovitz, "Optimal Control Theory," *Springer-Verlag,* New York, 1974.
2. J.D. Boissonnat, A. Cerezo and J. Leblond, "Shortest paths of bounded curvature in the plane," in *IEEE Int. Conf. on Robotics and Automation,* Nice, France, 1992.
3. J.D. Boissonnat, A. Cerezo and J. Leblond, "A Note on Shortest Paths in the Plane Subject to a Constraint on the Derivative of the Curvature," *INRIA Report No 2160, january 1994*
4. V.G. Boltyanskii, " Sufficient conditions for optimality and the justification of the dynamic programming method," *J.Siam Control,* vol 4, No 2, 1966.
5. R.W Brockett, " Control Theory and Singular Riemannian Geometry," *New Direction in Applied mathematics* (P.J. Hilton and G.S. Young, eds), Springer, pp 11-27, Berlin, 1981.
6. I.N. Bronhstein and K.A. Semendyayev, "A guide-book to Mathematics for technologists and engineers," *Pergamon Press,* 1964.
7. P. Brunovsky, "Every normal linear system has a regular synthesis," *J. Diff. Equations,* 28, pp 81-100, 1978.
8. P. Brunovsky, "Existence of regular synthesis for general problems," *J. Diff. Equations* 38 pp 317-343, 1980.
9. X-N. Bui, P. Souères, J-D. Boissonnat and J-P Laumond, "The Shortest Path Synthesis for Non-Holonomic Robots Moving Forwards", *INRIA Report N° 2153,* january 1994.
10. X-N. Bui, P. Souères, J-D. Boissonnat and J-P Laumond, "Shortest Path Synthesis for Dubins Nonholonomic Robot", *IEEE Int. Conf. on Robotics and Automation,* San Diego California, 1994.
11. X-N. Bui, "Planification de trajectoires pour un robot polygonal non-holonome dans un environement polygonal," *PhD Thesis, Ecole Nationale Supérieure des Mines de Paris, France* 1994.
12. S.S Cairns, "On the triangulation of regular loci", *Ann. of Math., 35, pp 579-587* (1934)
13. L. Cesari " Optimization, theory and applications," *Springer-Verlag,* New york, 1983.
14. R. Chatila, "Mobile robot navigation: space modeling and decisional processes," in *Robotics Research : The Third International Symposium* (O. Faugeras and G. Giralt, eds.), MIT Press, pp. 373–378, 1986.
15. E.J. Cockayne and G.W.C. Hall, "Plane motion of a particle subject to curvature constraints," *SIAM J. Control,* 13 (1), 1975.
16. L. E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics,* Vol. 79, pp. 497–516, 1957.
17. S. Fleury, P. Souères, J-P. Laumond, R. Chatila, " Primitives for Smoothing Mobile Robot Trajectories," *IEEE Transactions on Robotics and Automation* Vol. 11, No 3, June 1995.
18. M. Fliess, J. Levine, Ph. Martin, and P. Rouchon. " Sur les systèmes non linéaires différentiellement plats," *C.R. Acad. Sci Paris,* I-315:619-624, 1992.

19. J-Y.Fourquet "Mouvements en temps minimal pour les robots manipulateurs en tenant compte de leur dynamique non linéaire." *PhD Thesis, Université P.Sabatier* N° 800, France, 1990.
20. G. Giralt, R. Chatila, and M. Vaisset, "An integrated navigation and motion control system for autonomous multisensory mobile robots," in *Robotics Research : The First International Symposium* (M. Brady and R. P. Paul, eds.), MIT Press, pp. 191–214, 1984.
21. H. Halkin " Mathematical Foundation of System Optimization" in *Topics in Optimization,* edited by G. Leitmann, Academic Press, 1967.
22. P. Hartman. "The highway spiral for combining curves of different radii." *Trans. Amer. Soc. Civil Engin.,* 1957
23. A. Isidori, "Nonlinear Control Systems," (second edition) *Springer-Verlag,* 1989.
24. G. Jacob "Lyndon discretization and exact motion planning," in *European Control Conference,* pp. 1507-1512 Grenoble, France, 1991.
25. P. Jacobs, A Rege and J-P. Laumond, "Non-Holonomic Motion Planning for Hilare-Like Mobile Robots," *Proceedings of the International Symposium on intelligent Robotics,* Bangalore, 1991.
26. J.P. Laumond and P. Souères, "Metric induced by the shortest paths for a car-like mobile robot", in *IEEE IROS'93,* Yokohama, July 1993.
27. C. Lobry, "Controlabilité des systèmes non lineéaires," *outils et modèles mathématiques pour l'automatique, l'analyse des systèmes et le traitement du signal,* 1: pp 187-214, CNRS, 1981.
28. R. Felipe Monroy Pérez, "Non-Euclidian Dubin's problem: A control theoretic approach", *PhD thesis,* Department of Mathematics, University of Toronto, 1995.
29. L.S Pontryagin, V.G. Boltianskii, R.V. Gamkrelidze, and E.F. Mishenko. "The mathematical Theory of Optimal Processes," *Interscience Publishers,* 1962.
30. D.B. Reister and F.G. Pin, "Time-optimal trajectories for mobile robots with two independently driven wheels." *International Journal of Robotics Research, Vol 13, No 1, pp 38-54,* February 1994.
31. J. A. Reeds and R. A. Shepp, "Optimal paths for a car that goes both forwards and backwards, " *Pacific Journal of Mathematics,* 145 (2), 1990.
32. M. Renaud and Jean-Yves Fourquet, "Minimum-time motion of a mobile robot with two independant acceleration-driven wheels," *IEEE International Conference on Robotics and Automation,* Albuquerque, USA, April 1997.
33. P. Souères and J.P. Laumond, " Shortest paths synthesis for a car-like robot" *IEEE Transaction on Automatic Control,* Vol 41, No 5 May 1996.
34. P. Souères, "Comande optimale et robots mobiles non holonomes," *PhD Thesis, Universit'e Paul Sabatier, N° 1554,* France, 1993.
35. P. Souères, " Applying Boltianskii's sufficient optimality conditions to the characterization of shortest paths for the Reeds-Shepp car," *third European Control Conference ECC'95,* Roma, Italia, Sept. 1995.
36. H.J. Sussmann and W. Tang, "Shortest paths for the Reeds-Shepp car : a worked out example of the use of geometric techniques in nonlinear optimal control," *Report SYCON-91-10, Rutgers University,* 1991.
37. H.J. Sussmann, "Shortest 3-dimensional paths with a prescribed curvature bound", *Proc. of the 34th Conference on Decision and Control,* New Orleans, LA - December 1995.

38. H.J. Sussmann, "The Markov-Dubins problem with angular acceleration control", *Proc. of the 36th Conference on Decision and Control*, San Diego, CA - December 1997.
39. M.I. Zelikin and V.F. Borisov, "Theory of Chattering Control, with applications to astronotics, robotics, economics and engineering", *Birkhäuser*, Boston, 1994.

Feedback Control of a Nonholonomic Car-Like Robot

A. De Luca[1], G. Oriolo[1] and C. Samson[2]

[1] Università di Roma "La Sapienza"
[2] INRIA, Sophia-Antipolis

1 Introduction

The subject of this chapter is the control problem for nonholonomic wheeled mobile robots moving on the plane, and in particular the use of *feedback* techniques for achieving a given motion task.

In automatic control, feedback improves system performance by allowing the successful completion of a task even in the presence of external disturbances and/or initial errors. To this end, real-time sensor measurements are used to reconstruct the robot state. Throughout this study, the latter is assumed to be available at every instant, as provided by proprioceptive (e.g., odometry) or exteroceptive (sonar, laser) sensors.

We will limit our analysis to the case of a robot workspace free of obstacles. In fact, we implicitly consider the robot controller to be embedded in a hierarchical architecture in which a higher-level planner solves the obstacle avoidance problem and provides a series of motion goals to the lower control layer. In this perspective, the controller deals with the basic issue of converting ideal plans into actual motion execution. Wherever appropriate, we shall highlight the interactions between feedback control and motion planning primitives, such as the generation of open-loop commands and the availability of a feasible smooth path joining the current robot position to the destination.

The specific robotic system considered is a vehicle whose kinematic model approximates the mobility of a car. The configuration of this robot is represented by the position and orientation of its main body in the plane, and by the angle of the steering wheels. Two velocity inputs are available for motion control. This situation covers in a realistic way many of the existing robotic vehicles. Moreover, the *car-like* robot is the simplest nonholonomic vehicle that displays the general characteristics and the difficult maneuverability of higher-dimensional systems, e.g., of a car towing trailers. As a matter of fact, the control results presented here can be directly extended to more general kinematics, namely to all mobile robots admitting a chained-form representation. In particular, our choice encompasses the case of unicycle kinematics, another ubiquitous model of wheeled mobile robot, for which simple but specific feedback control methods can also be derived.

The nonholonomic nature of the car-like robot is related to the assumption that the robot wheels roll without slipping. This implies the presence of a nonintegrable set of first-order differential constraints on the configuration variables. While these nonholonomic constraints reduce the instantaneous motions that the robot can perform, they still allow global controllability in the configuration space. This unique feature leads to some challenging problems in the synthesis of feedback controllers, which parallel the new research issues arising in nonholonomic motion planning. Indeed, the wheeled mobile robot application has triggered the search for innovative types of feedback controllers that can be used also for more general nonlinear systems.

In the rest of this introduction, we present a classification of motion control problems, discussing their intrinsic difficulty and pointing out the relationships between planning and control aspects.

1.1 Problem classification

In order to derive the most suitable feedback controllers for each case, it is convenient to classify the possible motion tasks as follows:

- *Point-to-point motion*: The robot must reach a desired goal configuration starting from a given initial configuration.
- *Path following*: The robot must reach and follow a geometric path in the cartesian space starting from a given initial configuration (on or off the path).
- *Trajectory tracking*: The robot must reach and follow a trajectory in the cartesian space (i.e., a geometric path with an associated timing law) starting from a given initial configuration (on or off the trajectory).

The three tasks are sketched in Fig. 1, with reference to a car-like robot.

Using a more control-oriented terminology, the point-to-point motion task is a *stabilization* problem for an (equilibrium) point in the robot state space. For a car-like robot, two control inputs are available for adjusting four configuration variables, namely the two cartesian coordinates characterizing the position of a reference point on the vehicle, its orientation, and the steering wheels angle. More in general, for a car-like robot towing N trailers, we have two inputs for reconfiguring $n = 4 + N$ states. The error signal used in the feedback controller is the difference between the current and the desired configuration.

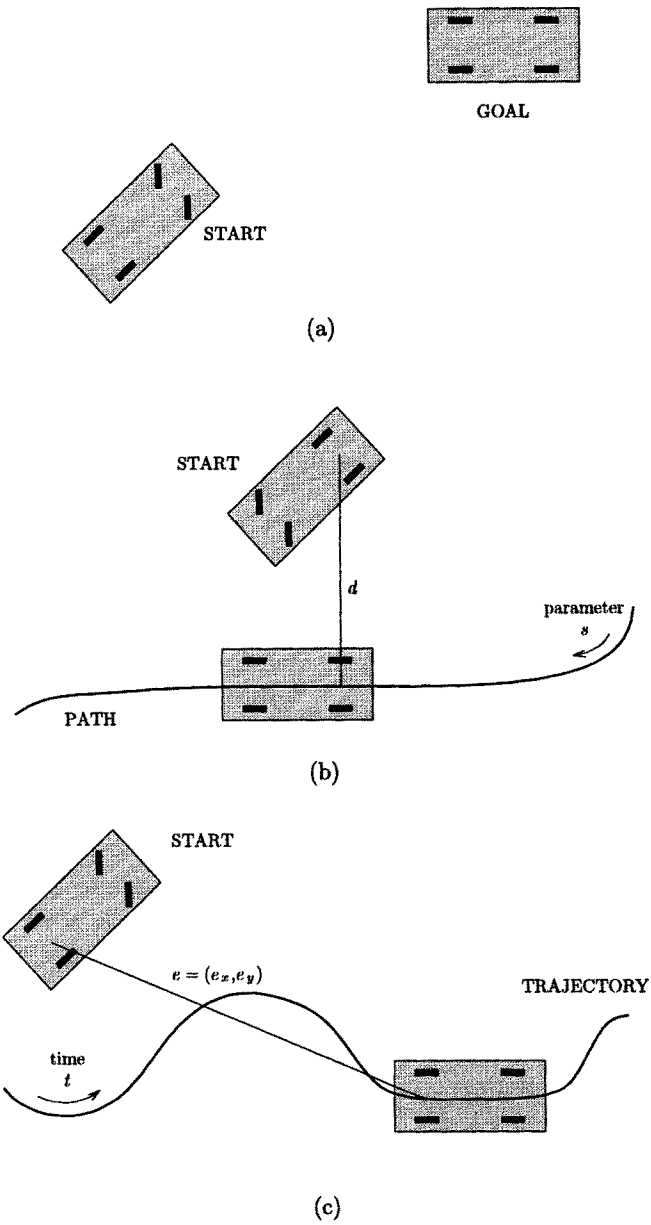


Fig. 1. Motion tasks: Point-to-point motion (a), Path following (b), Trajectory tracking (c)

In the path following task, the controller is given a geometric description of the assigned cartesian path. This information is usually available in a parameterized form expressing the desired motion in terms of a path parameter σ, which may be in particular the arc length along the path. For this task, time dependence is not relevant because one is concerned only with the geometric displacement between the robot and the path. In this context, the time evolution of the path parameter is usually free and, accordingly, the command inputs can be arbitrarily scaled with respect to time without changing the resulting robot path. It is then customary to set the robot forward velocity (one of the two inputs) to an arbitrary constant or time-varying value, leaving the second input available for control. The path following problem is thus rephrased as the stabilization to zero of a suitable scalar path error function (the distance d to the path in Fig. 1b) using only one control input. For the car-like robot, we shall see that achieving $d \equiv 0$ implies the control of three configuration variables—one less than the dimension of the configuration space—because higher-order derivatives of the controlled output d are related to these variables. Similarly, in the presence of N trailers, requiring $d \equiv 0$ involves the control of as many as $n - 1 = N + 3$ coordinates using one input.

In the trajectory tracking task, the robot must follow the desired cartesian path with a specified timing law (equivalently, it must track a moving reference robot). Although the trajectory can be split into a parameterized geometric path and a timing law for the parameter, such separation is not strictly necessary. Often, it is simpler to specify the workspace trajectory as the desired time evolution for the position of some representative point of the robot. The trajectory tracking problem consists then in the stabilization to zero of the two-dimensional cartesian error e (see Fig. 1c) using both control inputs. For the car-like robot, imposing $e \equiv 0$ over time implies the control of all four configuration variables. Similarly, in the presence of N trailers, we are actually controlling $n = N + 4$ coordinates using two inputs.

The point stabilization problem can be formulated in a local or in a global sense, the latter meaning that we allow for initial configurations that are arbitrarily far from the destination. The same is true also for path following and trajectory tracking, although locality has two different meanings in these tasks. For path following, a local solution means that the controller works properly provided we start sufficiently close to the path; for trajectory tracking, closeness should be evaluated with respect to the current position of the reference robot.

The amount of information that should be provided by a high-level motion planner varies for each control task. In point-to-point motion, information is reduced to a minimum (i.e., the goal configuration only) when a globally stabilizing feedback control solution is available. However, if the initial error is large, such a control may produce erratic behavior and/or large control effort

which are unacceptable in practice. On the other hand, a local feedback solution requires the definition of intermediate subgoals at the task planning level in order to get closer to the final desired configuration.

For the other two motion tasks, the planner should provide a path which is kinematically feasible (namely, which complies with the nonholonomic constraints of the specific vehicle), so as to allow its perfect execution in nominal conditions. While for an omnidirectional robot any path is feasible, some degree of geometric smoothness is in general required for nonholonomic robots. Nevertheless, the intrinsic feedback structure of the driving commands enables to recover transient errors due to isolated path discontinuities. Note also that the unfeasibility arising from a lack of continuity in some higher-order derivative of the path may be overcome by appropriate motion timing. For example, paths with discontinuous curvature (like the Reeds and Shepp optimal paths under maximum curvature constraint) can be executed by the real axle midpoint of a car-like vehicle provided that the robot is allowed to stop, whereas paths with discontinuous tangent are not feasible. In this analysis, the selection of the robot representative point for path/trajectory planning is critical.

The timing profile is the additional item needed in trajectory tracking control tasks. This information is seldom provided by current motion planners, also because the actual dynamics of the specific robot are typically neglected at this level. The above example suggests that it may be reasonable to enforce already at the planning stage requirements such as 'move slower where the path curvature is higher'.

1.2 Control issues

From a control point of view, the previously described motion tasks are defined for the nonlinear system

$$\dot{q} = G(q)v, \tag{1}$$

representing the kinematic model of the robot. Here, q is the n-vector of robot generalized coordinates, v is the m-vector of input velocities ($m < n$), and the columns g_i ($i = 1, \ldots, m$) of matrix G are smooth vector fields. For the car-like robot, it is $n = 4$ and $m = 2$.

The above model can be directly derived from the nonintegrable rolling constraints governing the system kinematic behavior. System (1) is driftless, a characteristic of first-order kinematic models. Besides, its nonlinear nature is intrinsically related to the nonholonomy of the original Pfaffian constraints. In turn, it can be shown that this is equivalent to the global accessibility of the n-dimensional robot configuration space—in spite of the reduced number of inputs.

Interestingly, the nonholonomy of system (1) reverses the usual order of difficulty of robot control tasks. For articulated manipulators, and in general for all mechanical systems with as many control inputs as generalized coordinates, stabilization to a fixed configuration is simpler than tracking a trajectory. Instead, stabilizing a wheeled mobile robot to a point is more difficult than path following or trajectory tracking.

A simple way to appreciate such a difference follows from the general discussion of the previous section. The point-to-point task is actually an input-state problem with $m = 2$ inputs and n controlled states. The path following task is an input-output problem with $m = 1$ input and $p = 1$ controlled output, implying the indirect control of $n - 1$ states. The trajectory tracking task is again an input-output problem with $m = 2$ inputs and $p = 2$ controlled outputs, implying the indirect control of n states. As a result, the point-to-point motion task gives rise to the most difficult control problem, since we are trying to control n independent variables using only two input commands. The path following and trajectory tracking tasks have a similar level of difficulty, being 'square' control problems (same number of control inputs and controlled variables).

This conclusion can be supported by a more rigorous controllability analysis. In particular, one can test whether the above problems admit an approximate solution in terms of linear control design techniques. We shall see that if the system (1) is linearized at a fixed configuration, the resulting linear system is not controllable. On the other hand, the linearization of eq. (1) about a smooth trajectory gives rise to a linear time-varying system that is controllable, provided some persistency conditions are satisfied by the reference trajectory.

The search for a feedback solution to the point stabilization problem is further complicated by a general theoretical obstruction. Although the kinematic model (1) can be shown to be controllable using nonlinear tools from differential geometry, it fails to satisfy a necessary condition for stabilizability via smooth time-invariant feedback (Brockett's theorem). This means that the class of stabilizing controllers should be suitably enlarged so as to include nonsmooth and/or time-varying feedback control laws.

We finally point out that the design of feedback controllers for the path following task can be tackled from two opposite directions. In fact, by separating the geometric and timing information of a trajectory, path following may be seen as a subproblem of trajectory tracking. On the other hand, looking at the problem from the point of view of controlled states (in the proper coordinates), path following appears as part of a point stabilization task. The latter philosophy will be adopted in this chapter.

1.3 Open-loop vs. closed-loop control

Some comments are now appropriate concerning the relationships between the planning and control phases in robot motion execution.

Essentially, we regard planning and open-loop (or feedforward) control as synonyms, as opposed to feedback control. In a general setting, a closed-loop controller results from the superposition of a feedback action to a coherent feedforward term. The latter is determined based on a priori knowledge about the motion task and the environment, which may have been previously acquired by exteroceptive sensors. Feedback control is instead computed in real-time based on external/internal sensor data.

However, the borderline between open-loop and closed-loop control solutions may not be so sharp. In fact, we may use repeated open-loop phases, replanned at higher rates using new sensor data to gather information on the actual state. In the limit, continuous sensing and replanning leads to a feedback solution. Although this scheme is conceptually simple, its convergence analysis may not be easy. Thus, we prefer to consider the planning and control phases separately.

For wheeled mobile robots, the usual output of the planning phase, which takes into account the obstacle avoidance requirement, is a kinematically feasible path with associated nominal open-loop commands. To guarantee feasibility, the planner may either take directly into account the nonholonomic constraints in the generation of a path, or create a preliminary holonomic path with standard techniques and then approximate it with a concatenation of feasible subpaths.

In the planning phase, it is also possible to include an optimality criterion together with system state and input constraints. It is often possible to obtain a solution by applying optimal (open-loop) control results. A typical cost criterion for the car-like robot is the total length of the collision-free path joining source to destination, while constraints include bounds on the steering angle as well as on the linear and angular velocity. In any case, the resulting commands are computed off-line. Hence, unmodeled events at running time, such as occasional slipping of the wheels or erroneous initial localization, will prevent the successful completion of a point-to-point motion or the correct tracing of a desired path.

The well-known answer to such problems is resorting to a feedback controller, driven by the current task error, so as to achieve some degree of robustness. However, this should by no means imply the abdication to the use of the nominal open-loop command computed in the planning phase, which is included as the feedforward term in the closed-loop controller. As soon as the task error is zero, the feedback signal is not in action and the output command of the controller coincides with the feedforward term.

The path and trajectory tracking controllers presented in this chapter agree with this combined approach. In fact, the feedforward represents the anticipative action needed to drive the robot along the desired nominal motion. We point out that a shortcoming arises when the planner generates optimal feedforward commands that are at their saturation level, because this leaves no room for the correcting feedback action. This is a common problem in open-loop optimal control; unfortunately, optimal feedback control laws for nonlinear systems are quite difficult to obtain in explicit form.

On the other hand, it follows from the discussion in Sect. 1.1 that no feedforward is required in principle for the point stabilization task, so that the executed trajectory results from the feedback action alone. While this approach may be satisfactory for fine motion tasks, in gross motion a pure feedback control may drive the mobile robot toward the goal in an unpredictable way. In this case, a closer integration of planning and control would certainly improve the overall performance.

1.4 Organization of contents

We will present some of the most significant feedback control strategies for the different robot motion tasks. For each method, we discuss the single design steps and illustrate the typical performance by simulations. Results are presented in a consistent way in order to allow for comparisons. The organization of the rest of the chapter is as follows.

Section 2 is devoted to preliminary material. The kinematic model of the car-like robot is introduced, stating the main assumptions and distinguishing the cases of rear-wheel and front-wheel driving. We analyze the local controllability properties at a configuration and about a trajectory. Global controllability is proved in a nonlinear setting and a negative result concerning smooth feedback stabilizability is recalled. This section is concluded by presenting the chained-form transformation of the model and its essential features.

In Sect. 3 we address the trajectory tracking problem. The generation of suitable feedforward commands for a given smooth trajectory is discussed. In particular, we point out how geometric and timing information can be handled separately. A simple linear controller is devised for the chained-form representation of the car-like robot, using the approximate system linearization around the nominal trajectory. Then, we present two nonlinear controllers based on exact feedback linearization. The first uses static feedback to achieve input-output linearization for the original kinematic model of the car-like robot. The second is a full-state linearizing dynamic feedback designed on the chained-form representation. Both guarantee global tracking with prescribed linear error dynamics.

In Sect. 4 two time-varying feedback control laws are presented, both solving the point stabilization as well as the path following problem. The two controllers are globally defined on chained-form representations. The first is a smooth time-varying controller based on a Lyapunov analysis of skew-symmetric forms. The second is a nonsmooth time-varying feedback controller inspired by the backstepping approach. Convergence rates of the two methods are discussed and illustrated by simulations.

Section 5 summarizes the obtained results and indicates some possible extensions of the control problem to address the limitations arising in real-world problems.

In the exposition, we shall limit the references only to the basic sources from which the presented material is drawn. In the concluding section, however, a reasoned guide to further related literature is given.

2 Modeling and analysis of the car-like robot

In this section, we shall first derive the kinematic equations of a car-like robot and then analyze the fundamental properties of the corresponding system from a control viewpoint.

2.1 Kinematic modeling

The main feature of the kinematic model of wheeled mobile robots is the presence of nonholonomic constraints due to the *rolling without slipping* condition between the wheels and the ground. The case of a single wheel is analyzed first.

Consider a wheel that rolls on a plane while keeping its body vertical, as shown in Fig. 2. This kind of system is also referred to as a *unicycle*. Its configuration can be described by a vector q of three generalized coordinates, namely the position coordinates x, y of the point of contact with the ground in a fixed frame and the angle θ measuring the wheel orientation with respect to the x axis. The system generalized velocities $\dot{q}$ cannot assume independent values; in particular, they must satisfy the constraint

$$\begin{bmatrix} \sin\theta & -\cos\theta & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = 0, \tag{2}$$

entailing that the linear velocity of the wheel center lies in the body plane of the wheel (zero lateral velocity).

Equation (2) is a typical example of *Pfaffian* constraint $C(q)\dot{q} = 0$, i.e., linear in the generalized velocities. As a consequence, all admissible generalized

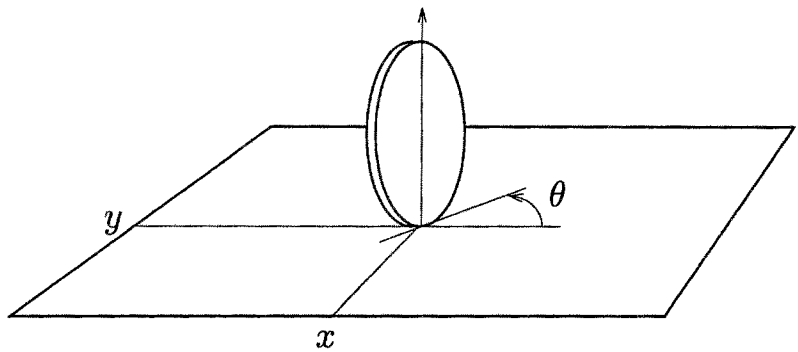


Fig. 2. Generalized coordinates of a unicycle

velocities are contained in the null space of the constraint matrix $C(q)$. In this case, one obtains

$$\dot{q} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} v_2, \tag{3}$$

where v_1 and v_2 are respectively the linear velocity of the wheel and its angular velocity around the vertical axis. As the choice of a basis for the null space of matrix C is not unique, the components of v may also assume different meanings. Moreover, they may have no direct relationship with the actual controls available, that are in general forces or torques. For this reason, eq. (3) is called the *kinematic model* of the unicycle.

Let us now turn to a robot having the same kinematics of an automobile, as shown in Fig. 3. For simplicity, assume that the two wheels on each axle (front and rear) collapse into a single wheel located at the midpoint of the axle (*car-like* model). The front wheel can be steered while the rear wheel orientation is fixed. The generalized coordinates are $q = (x, y, \theta, \phi)$, where x, y are the cartesian coordinates of the rear wheel, θ measures the orientation of the car body with respect to the x axis, and ϕ is the steering angle.

The system is subject to two nonholonomic constraints, one for each wheel:

$$\begin{aligned} \dot{x}_f \sin(\theta + \phi) - \dot{y}_f \cos(\theta + \phi) &= 0 \\ \dot{x} \sin\theta - \dot{y} \cos\theta &= 0, \end{aligned}$$

with x_f, y_f denoting the cartesian coordinates of the front wheel. By using the rigid-body constraint

$$x_f = x + \ell \cos\theta$$

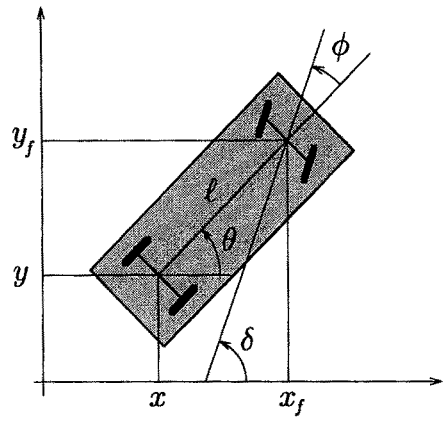


Fig. 3. Generalized coordinates of a car-like robot

$$y_f = y + \ell \sin\theta,$$

where ℓ is the distance between the wheels, the first kinematic constraint becomes

$$\dot{x}\sin(\theta+\phi) - \dot{y}\cos(\theta+\phi) - \dot{\theta}\,\ell\cos\phi = 0.$$

The Pfaffian constraint matrix is

$$C(q) = \begin{bmatrix} \sin(\theta+\phi) & -\cos(\theta+\phi) & -\ell\cos\phi & 0 \\ \sin\theta & -\cos\theta & 0 & 0 \end{bmatrix},$$

and has constant rank equal to 2.

If the car has *rear-wheel driving*, the kinematic model is derived as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/\ell \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2, \tag{4}$$

where v_1 and v_2 are the *driving* and the *steering* velocity input, respectively. There is a model singularity at $\phi = \pm\pi/2$, where the first vector field has a discontinuity. This corresponds to the car becoming jammed when the front wheel is normal to the longitudinal axis of the body. However, the importance of this singularity is limited, due to the restricted range of the steering angle ϕ in most practical cases.

The model for *front-wheel driving* is obtained as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi/\ell \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2,$$

where the driving velocity v_1 refers now to the front wheel. Note that the previous singularity does not occur in this model; in fact, at $\phi = \pm\pi/2$ the car can still (in principle) pivot about its rear wheel.

An interesting format for the kinematic equations of a front-wheel drive car can be obtained by means of a change of coordinates and an input transformation. In particular, define the alternative set of generalized coordinates $(x_f, y_f, \delta, \theta)$, where $\delta = \theta + \phi$ is the *absolute* steering angle with respect to the x-axis (see Fig. 3). By using the input transformation

$$\begin{aligned} w_1 &= v_1 \\ w_2 &= \frac{1}{\ell}\sin(\delta - \theta)v_1 + v_2, \end{aligned}$$

it is easy to show that

$$\begin{bmatrix} \dot{x}_f \\ \dot{y}_f \\ \dot{\delta} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\delta \\ \sin\delta \\ 0 \\ \sin(\delta - \theta)/\ell \end{bmatrix} w_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} w_2.$$

Note that the first three equations are those of a unicycle. As a matter of fact, the above model is equivalent to a unicycle with one trailer hooked at the center of the wheel. Correspondingly, the new input w_2 is the absolute (i.e., measured w.r.t. the x axis) steering velocity of the front wheel. Other state and input transformations of the car-like kinematics will be presented in Sect. 2.3.

Throughout the rest of this chapter, we shall be dealing with the rear-wheel drive model (4). It has to be mentioned that a more complete kinematic model should include also the rotation angles of each wheel as generalized coordinates, in order to account for the presence of actuators and sensors on the wheel axis as well as for typical nonidealities such as tire deformation. Nevertheless, our model captures the essence of the vehicle kinematics and is well suited for control purposes.

2.2 Controllability analysis

Equation (4) may be rewritten as

$$\dot{q} = g_1(q)v_1 + g_2(q)v_2, \qquad \text{with} \quad g_1 = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/\ell \\ 0 \end{bmatrix}, \; g_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{5}$$

The above system is nonlinear, driftless (i.e., no motion takes place under zero input) and there are less control inputs than generalized coordinates.

Although any driver's experience indicates that a car-like robot should be completely controllable, it is not trivial to establish such property on a mathematical basis. In particular, we shall see that an approximate linear analysis is not sufficient in general to achieve this goal.

Controllability at a point As system (5) is driftless, any configuration q_e is an equilibrium point under zero input. The easiest way to investigate its controllability at q_e is to consider the corresponding linear approximation

$$\dot{\tilde{q}} = g_1(q_e)v_1 + g_2(q_e)v_2 = G(q_e)v,$$

where $\tilde{q} = q - q_e$. The rank of the controllability matrix $G(q_e)$ is two. Hence, the linearized system is not controllable so that a linear controller will not work, not even locally.

A useful tool that allows to test the controllability of driftless nonlinear systems is the *Lie Algebra rank condition* [18]. In our case, this boils down to check whether

$$\operatorname{rank}[g_1 \;\; g_2 \;\; [g_1, g_2] \;\; [g_1, [g_1, g_2]] \;\; [g_2, [g_1, g_2]] \cdots \;] = 4.$$

For system (5), the first two Lie brackets are computed as

$$[g_1, g_2] = \begin{bmatrix} 0 \\ 0 \\ -1/\ell\cos^2\phi \\ 0 \end{bmatrix}, \qquad [g_1, [g_1, g_2]] = \begin{bmatrix} -\sin\theta/\ell\cos^2\phi \\ \cos\theta/\ell\cos^2\phi \\ 0 \\ 0 \end{bmatrix}.$$

It is easy to verify that, away from the model singularity $\phi = \pm\pi/2$, the above rank is 4, so that the car-like robot is certainly controllable whenever the steering angle is different from $\pm\pi/2$. Using the fact that ϕ can be modified at will through the control input v_2, it can be shown that the system is actually controllable everywhere.

As for the stabilizability of system (5), the failure of the previous linear analysis indicates that exponential stability in the sense of Lyapunov cannot be achieved by smooth feedback [45]. However, things turn out to be even worse: it is not possible to stabilize at all the system at q_e by using a smooth (in fact, continuous) time-invariant feedback law $v = v(q)$. This negative result can be readily established on the basis of Brockett's theorem [6], which implies that a necessary condition for smooth stabilizability of a driftless *regular* system (i.e., such that the input vector fields are linearly independent at q_e) is that the number of inputs equals the number of states. Since this is not the case, such condition is violated.

The above limitation has a deep impact on the control design approach. To obtain a point stabilizing controller it is either necessary to give up the continuity requirement, or to resort to time-varying control laws $v = v(q, t)$. In Sect. 4 we shall pursue the latter approach.

Controllability about a trajectory Consider now a desired reference state trajectory $q_d(t) = (x_d(t), y_d(t), \theta_d(t), \phi_d(t))$ for the car-like robot. In order to be feasible, this trajectory must satisfy the nonholonomic constraints on the vehicle motion. The generation of such trajectories as well as of the corresponding reference velocity inputs v_{d1} and v_{d2} will be addressed in Sect. 3.

Defining $\tilde{q}(t) = q(t) - q_d(t)$ and $\tilde{v}(t) = v(t) - v_d(t)$, the approximate linearization of system (5) about the reference trajectory is obtained as

$$\dot{\tilde{q}} = A(t)\tilde{q} + B(t)\tilde{v}, \tag{6}$$

with

$$A(t) = \sum_{i=1}^{2} v_{di}(t) \frac{\partial g_i}{\partial q}\bigg|_{q=q_d(t)}, \qquad B(t) = G(q_d(t)).$$

Simple computations yield

$$A(t) = \begin{bmatrix} 0 & 0 & -\sin\theta_d(t) v_{d1}(t) & 0 \\ 0 & 0 & \cos\theta_d(t) v_{d1}(t) & 0 \\ 0 & 0 & 0 & v_{d1}(t)/\ell \cos^2\theta_d(t) \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B(t) = \begin{bmatrix} \cos\theta_d(t) & 0 \\ \sin\theta_d(t) & 0 \\ \tan\theta_d(t)/\ell & 0 \\ 0 & 1 \end{bmatrix}.$$

Note that the linearized system is *time-varying* through the dependence on time of the reference trajectory. As a consequence, the controllability analysis is more involved than in the time-invariant case, and would consist in testing whether the *controllability Gramian* is nonsingular [19].

For illustration, we consider the special case of a linear trajectory with constant velocity, in which one falls upon a time-invariant system. In fact, in

this situation we have $v_{d1}(t) \equiv v_{d1}$ (a constant nonzero value) and $v_{d2}(t) \equiv 0$. Besides, $\theta_d(t) \equiv \theta_d(t_0)$ and $\phi(t) \equiv 0$. The controllability condition is

$$\text{rank}\left[B \;\; AB \;\; A^2B \;\; A^3B\right] = 4.$$

It is easy to verify that the controllability matrix has a single nonzero 4×4 minor whose value is $-u_{d1}^3/\ell^2 \cos^4\theta_d$. Therefore, the linearized system is controllable as long as $\theta_d \neq \pm\pi/2$ and $u_{d1} \neq 0$ (which is not unexpected, since for $u_{d1} = 0$ the trajectory would collapse to a point). This implies that system (5) can be locally stabilized about the reference trajectory by a linear feedback.

Although the above analysis has been carried out for a linear reference trajectory, we shall see in Sect. 3 that it is possible to design a locally stabilizing linear feedback for arbitrary feasible trajectories provided they do not come to a stop.

2.3 Chained forms

The existence of canonical forms for kinematic models of nonholonomic robots is essential for the systematic development of both open-loop and closed-loop control strategies. The most useful canonical structure is the *chained form.*

The two-input driftless control system

$$\begin{aligned} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_2 u_1 \\ &\vdots \\ \dot{x}_n &= x_{n-1} u_1, \end{aligned} \tag{7}$$

is called $(2, n)$ single-chain form [28]. The two-input case covers many of the kinematic models of practical wheeled mobile robots. A more general study would involve multiple chains, with rather straightforward extensions of the results presented in this chapter.

The chained system (7), although nonlinear, has a strong underlying linear structure. This clearly appears when u_1 is assigned as a function of time, and is no longer regarded as a control variable. In this case, eq. (7) becomes a single-input, time-varying linear system.

The $(2, n)$ chained form can be shown to be completely controllable by application of the Lie Algebra rank condition. In performing this calculation, one finds that all Lie brackets above a certain order (namely, $n - 2$) are identically zero; this property of the system is called *nilpotency.*

Necessary and sufficient conditions have been given in [29] for the conversion of a two-input system like (5) into chained form by means of *(i)* a change of

coordinates $x = \phi(q)$, and *(ii)* an invertible input transformation $v = \beta(q)u$. By applying these conditions, one can show that nonholonomic systems with $m = 2$ inputs and $n = 3$ or 4 generalized coordinates can be always put in chained form.

For example, consider the kinematic model (3) of a unicycle. By letting

$$\begin{aligned} x_1 &= -\theta \\ x_2 &= x\cos\theta + y\sin\theta \\ x_3 &= -x\sin\theta + y\cos\theta, \end{aligned}$$

and

$$\begin{aligned} v_1 &= x_3 u_1 + u_2 = (-x\sin\theta + y\cos\theta)u_1 + u_2 \\ v_2 &= -u_1, \end{aligned}$$

it is easy to see that the transformed system is in (2,3) chained form. Besides, both the coordinate and the input transformation are globally defined. Note that the new variables x_2 and x_3 are simply the cartesian coordinates of the unicycle evaluated in the mobile frame attached to the robot body and rotated so as to align the x_2 axis with the vehicle orientation.

Let us now consider the car-like robot model (5). Using the change of coordinates

$$\begin{aligned} x_1 &= x \\ x_2 &= \tan\phi/\ell\cos^3\theta \\ x_3 &= \tan\theta \\ x_4 &= y, \end{aligned} \tag{8}$$

together with the input transformation

$$\begin{aligned} v_1 &= u_1/\cos\theta \\ v_2 &= -3\sin\theta\sin^2\phi\, u_1/\ell\cos^2\theta + \ell\cos^3\theta\cos^2\phi\, u_2, \end{aligned} \tag{9}$$

the system is in $(2,4)$ chained form

$$\begin{aligned} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_2 u_1 \\ \dot{x}_4 &= x_3 u_1. \end{aligned} \tag{10}$$

In this case, the transformation (and thus, the obtained chained form) is only locally defined in open connected domains which exclude the vehicle orientations $\theta = \pi/2 \pm k\pi$, $k \in \mathbb{N}$. The structure of change of coordinates (8) is

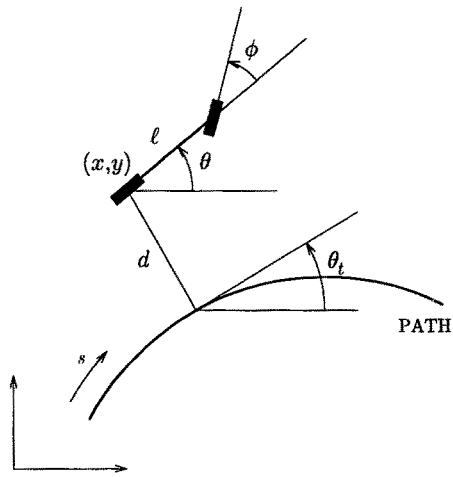


Fig. 4. Coordinate definition for a path following task

interesting because it can be generalized to nonholonomic systems of higher dimension, such as the N-trailer robot [46]. In particular, the x_1 and x_n coordinates can be always chosen as the x and y coordinates of the midpoint of the last trailer wheel axle.

It is interesting to note that the $(2, n)$ chained form can be also obtained starting from a different point of view. In particular, assume that the car-like robot must follow a given path which is parameterized by its arc length. With reference to Fig. 4, let d be the distance between the rear axle midpoint and the path, and s be the corresponding value of the path parameter. Denote by θ_t the angle between the current tangent to the path and the x axis, and let $\theta_p = \theta - \theta_t$. The curvature along the path is defined as

$$c(s) = \frac{d\theta_t}{ds},$$

which implies

$$\dot{\theta}_t = c(s)\dot{s}. \tag{11}$$

In the following, we assume that $c(\cdot) \in C^1$ and that the path satisfies some technical conditions (see [44] for details). It is easy to verify that

$$\dot{s} = v_1 \cos\theta_p + \dot{\theta}_t\, d \tag{12}$$

$$\dot{d} = v_1 \sin\theta_p. \tag{13}$$

Equations (12–13) may be combined with model (4) in order to derive the kinematic equations in terms of *path coordinates* $q_p = (s, d, \theta_p, \phi)$:

$$\begin{bmatrix} \dot{s} \\ \dot{d} \\ \dot{\theta}_p \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \dfrac{\cos\theta_p}{1 - dc(s)} \\ \sin\theta_p \\ \left(\dfrac{\tan\phi}{\ell} - \dfrac{c(s)\cos\theta_p}{1 - dc(s)}\right) \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2. \tag{14}$$

The above model can be put in the (2, 4) chained form by using the change of coordinates

$$\begin{aligned} x_1 &= s \\ x_2 &= -c'(s)d\tan\theta_p - c(s)(1 - dc(s))\,\frac{1 + \sin^2\theta_p}{\cos^2\theta_p} + \frac{(1 - dc(s))^2\tan\phi}{\ell\cos^3\theta_p} \\ x_3 &= (1 - dc(s))\tan\theta_p \\ x_4 &= d, \end{aligned} \tag{15}$$

together with the input transformation

$$\begin{aligned} v_1 &= \frac{1 - dc(s)}{\cos\theta_p}\,u_1 \\ v_2 &= \alpha_2(q_p)(u_2 - \alpha_1(q_p)u_1). \end{aligned}$$

In the above formulas, $c'(s)$ denotes the derivative of c with respect to s, and we have set

$$\begin{aligned} \alpha_1(q_p) &= \frac{\partial x_2}{\partial s} + \frac{\partial x_2}{\partial d}\,(1 - dc(s))\tan\theta_p + \frac{\partial x_2}{\partial \theta_p}\left(\frac{\tan\phi\,(1 - dc(s))}{\ell\cos\theta_p} - c(s)\right) \\ \alpha_2(q_p) &= \frac{\ell\cos^3\theta_p\cos^2\phi}{(1 - dc(s))^2}. \end{aligned}$$

Also this chained-form transformation is locally defined in open connected domains, because $\theta_p = \pi/2 \pm k\pi$, $k \in \mathbb{N}$, must be excluded. Note that in the particular case $c(s) \equiv 0$, one recovers the previous expressions (8) and (9). In fact, in this situation the path may be taken as the x-axis of the world frame, and (s, d, θ_p) become the coordinates (x, y, θ) of the vehicle.

We conclude this section by pointing out that there are other canonical forms that can be successfully used in connection with nonholonomic systems, namely the *Čaplygin form* and the *power form*. It is noteworthy that, for $m = 2$ inputs, the three canonical forms are mathematically equivalent, since there exist global coordinate transformations that allow to convert one into the others [21].

3 Trajectory tracking

In this section, we consider the problem of tracking a given cartesian trajectory with the car-like robot using feedback control. Both a local and a global approach will be presented. In the first, we use a standard linear control design and obtain convergence provided that the car starts sufficiently close to the desired trajectory. In the second, we pursue a feedback linearization approach, achieving asymptotic stability for arbitrary initial states via static as well as dynamic nonlinear feedback.

In the following, extensive use is made of the chained-form representation. Such system transformation is not strictly necessary, but simplifies considerably the control design and provides at the same time a framework for the direct extension of the controllers to vehicles with more complex kinematics. In any case, the methods presented here can be applied to more general mobile robots, even those which cannot be put in chained form.

Before moving to the control design, we discuss the problem of generating state reference trajectories for the car-like robot, both in the original kinematic description (5) and in the chained form (10).

3.1 Reference trajectory generation

Assume that a feasible and smooth desired *output trajectory* is given in terms of the cartesian position of the car rear wheel, i.e.,

$$x_d = x_d(t), \quad y_d = y_d(t), \qquad t \geq t_0. \tag{16}$$

This natural way of specifying the motion of a car-like robot has an appealing property. In fact, from this information we are able to derive the corresponding time evolution of the remaining coordinates (*state trajectory*) as well as of the associated input commands (*input trajectory*) as shown hereafter.

The desired output trajectory (16) is feasible when it can be obtained from the evolution of a reference car-like robot

$$\dot{x}_d = \cos\theta_d \, v_{d1} \tag{17}$$

$$\dot{y}_d = \sin\theta_d \, v_{d1} \tag{18}$$

$$\dot{\theta}_d = \tan\phi_d \, v_{d1}/\ell \tag{19}$$

$$\dot{\phi}_d = v_{d2}, \tag{20}$$

for suitable initial conditions $(x_d(t_0), y_d(t_0), \theta_d(t_0), \phi_d(t_0))$ and piecewise-continuous inputs $v_d(t)$, for $t \geq t_0$.

Solving for v_{d1} from eqs. (17) and (18) gives for the first input

$$v_{d1}(t) = \pm\sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)}, \tag{21}$$

where the sign depends on the choice of executing the trajectory with forward or backward car motion, respectively.

Dividing eq. (18) by (17), and keeping the sign of the linear velocity input into account, we compute the desired orientation of the car as

$$\theta_d(t) = \text{ATAN2}\left\{\frac{\dot{y}_d(t)}{v_{d1}(t)}, \frac{\dot{x}_d(t)}{v_{d1}(t)}\right\}, \tag{22}$$

with codomain in all four quadrants.

Differentiating eqs. (17) and (18), and combining the results so as to eliminate $\dot{v}_{d1}$, we obtain

$$\dot{\theta}_d(t) = \frac{\ddot{y}_d(t)\dot{x}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{v_{d1}^2(t)}.$$

Plugging this into eq. (19) provides the desired steering angle

$$\phi_d(t) = \arctan\frac{\ell\left[\ddot{y}_d(t)\dot{x}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)\right]}{v_{d1}^3(t)}, \tag{23}$$

which takes values in $(-\pi/2, \pi/2)$.

Finally, differentiating (23) and substituting the result in eq. (20) yields the second input

$$v_{d2}(t) = \ell v_{d1}\frac{\left(\dddot{y}_d\dot{x}_d - \dddot{x}_d\dot{y}_d\right)v_{d1}^2 - 3\left(\ddot{y}_d\dot{x}_d - \ddot{x}_d\dot{y}_d\right)\left(\dot{x}_d\ddot{x}_d + \dot{y}_d\ddot{y}_d\right)}{v_{d1}^6 + \ell^2\left(\ddot{y}_d\dot{x}_d - \ddot{x}_d\dot{y}_d\right)^2}, \tag{24}$$

where we dropped for compactness the time dependence in the right hand side.

Equations (21–24) provide the *unique* state and input trajectory (modulo the choice of forward or backward motion) needed to reproduce the desired output trajectory. These expressions depend only on the values of the output trajectory (16) and its derivatives up to the third order. Therefore, in order to guarantee its exact reproducibility, the cartesian trajectory should be three times differentiable almost everywhere. This fact should be taken into account at the motion planning level.

For illustration, consider a circular trajectory of radius R to be traced counterclockwise at a constant linear velocity $R\omega$, with the rear wheel of the car located at the origin at time $t_0 = 0$. We have

$$x_d(t) = R\sin\omega t, \qquad y_d(t) = R(1 - \cos\omega t).$$

From the previous formulas, when the robot is required to move in the forward direction, the nominal command inputs are computed as

$$v_{d1}(t) = R\omega, \qquad v_{d2}(t) = 0,$$

while the unique initial state enabling exact reproduction of the desired trajectory is

$$x_d(0) = 0, \quad y_d(0) = 0, \quad \theta_d(0) = 0, \quad \phi_d(0) = \arctan \frac{\ell}{R}.$$

The only situation in which the reference signals (21–24) are not defined is when $v_{d1}(\bar{t}) = 0$ for some $\bar{t} \geq t_0$. In this case, it is convenient to use a parameterized trajectory in which the geometric path description is separated from the timing information. Denoting by σ the path parameter (e.g., the arc length s) and by $\sigma = \sigma(t)$ the time history along the trajectory, one has

$$\dot{x}_d(t) = \frac{d}{d\sigma} x_d(\sigma) \cdot \frac{d\sigma}{dt} = x_d'(\sigma(t))\dot{\sigma}(t)$$

and a similar expression for $\dot{y}_d(t)$ (the prime denotes differentiation with respect to the path parameter). We can then rewrite the linear pseudo-velocity command at a given point on the path as

$$w_{d1}(\sigma) = \pm\sqrt{x_d'^2(\sigma) + y_d'^2(\sigma)}. \tag{25}$$

The actual linear velocity input is expressed as

$$v_{d1}(t) = w_{d1}(\sigma(t))\dot{\sigma}(t).$$

The situation $v_{d1}(\bar{t}) = 0$ is then obtained by letting $\dot{\sigma}(\bar{t}) = 0$, with $w_{d1}(\sigma(\bar{t})) \neq 0$.

The desired orientation is computed as

$$\theta_d(\sigma) = \text{ATAN2}\left\{\frac{y_d'(\sigma)}{w_{d1}(\sigma)}, \frac{x_d'(\sigma)}{w_{d1}(\sigma)}\right\},$$

which is now always well defined. By performing the time/space separation also in eqs. (23) and (24), the zero-velocity singularity is similarly avoided, because only curvature and higher-order information about the path appear in the expressions of $\phi_d(\sigma)$ and $w_{d2}(\sigma)$, with $v_{d2}(t) = w_{d2}(\sigma(t))\dot{\sigma}(t)$. We have in fact

$$\phi_d(\sigma) = \arctan \frac{\ell\left(y_d''(\sigma)x_d'(\sigma) - x_d''(\sigma)y_d'(\sigma)\right)}{w_{d1}^3(\sigma)}$$

and

$$w_{d2}(\sigma) = \frac{\ell w_{d1}\left[\left(y_d''' x_d' - x_d''' y_d'\right) - 3\left(y_d'' x_d' - x_d'' y_d'\right)\left(x_d' x_d'' + y_d' y_d''\right)\right)}{w_{d1}^6 + \ell^2\left(y_d'' x_d' - x_d'' y_d'\right)^2},$$

where we dropped the dependence on σ for compactness.

The derivation of the reference inputs that generate a desired cartesian trajectory of the car-like robot can also be performed for the (2,4) chained form. In fact, with the reference system given by

$$\begin{aligned} \dot{x}_{d1} &= u_{d1} \\ \dot{x}_{d2} &= u_{d2} \\ \dot{x}_{d3} &= x_{d2} u_{d1} \\ \dot{x}_{d4} &= x_{d3} u_{d1}, \end{aligned} \tag{26}$$

from the output trajectory (16) and the change of coordinates (8) we easily obtain

$$\begin{aligned} x_{d1}(t) &= x_d(t) \\ x_{d2}(t) &= \left[\ddot{y}_d(t)\dot{x}_d(t) - \dot{y}_d(t)\ddot{x}_d(t)\right] / \dot{x}_d^3(t) \\ x_{d3}(t) &= \dot{y}_d(t)/\dot{x}_d(t) \\ x_{d4}(t) &= y_d(t), \end{aligned}$$

and

$$\begin{aligned} u_{d1}(t) &= \dot{x}_d(t) \\ u_{d2}(t) &= \left[\dddot{y}_d(t)\dot{x}_d^2(t) - \ddot{x}_d(t)\dot{y}_d(t)\dot{x}_d(t) - 3\ddot{y}_d(t)\dot{x}_d(t)\ddot{x}_d(t) + 3\dot{y}_d(t)\ddot{x}_d^2(t)\right] / \dot{x}_d^4(t). \end{aligned}$$

To work out an example for this case, consider a sinusoidal trajectory stretching along the x axis and starting from the origin at time $t_0 = 0$

$$x_d(t) = t, \qquad y_d(t) = A \sin \omega t. \tag{27}$$

The feedforward commands for the chained-form representation are given by

$$u_{d1}(t) = 1, \qquad u_{d2}(t) = -A\omega^3 \cos \omega t,$$

while its initial state should be set at

$$x_{d1}(0) = 0, \quad x_{d2}(0) = 0, \quad x_{d3}(0) = A\omega, \quad x_{d4}(0) = 0.$$

We note that, if the change of coordinates (8) is used, there is an 'asymmetric' singularity in the state and input trajectory when $\dot{x}_d(\bar{t}) = 0$, for some $\bar{t} > t_0$. This coincides with the situation $\theta_d(\bar{t}) = \pi/2$, where the chained-form transformation is not defined.

On the other hand, if the chained form comes from the model in path variables (14) through the change of coordinates (15), the state and input trajectory needed to track the reference output trajectory $s = s_d(t)$, $d = d_d(t) = 0$, $t \geq t_0$,

are simply obtained as

$$x_{d1}(t) = s_d(t)$$
$$x_{d2}(t) = 0$$
$$x_{d3}(t) = 0$$
$$x_{d4}(t) = 0$$

and

$$u_{d1}(t) = \dot{s}_d(t)$$
$$u_{d2}(t) = 0,$$

without any singularity.

Similar developments can be repeated more in general, e.g., for the case of a nonholonomic mobile robot with N trailers. In fact, once the position of the last trailer is taken as the system output, it is possible to compute the evolution of the remaining state variables as well as of the system inputs as functions of the output trajectory (i.e., of the output and its derivatives up to a certain order). Not surprisingly, the same is true for the chained form (7) by defining (x_1, x_n) as system outputs.

The above property has been also referred to as *differential flatness* [36], and is mathematically equivalent to the existence of a dynamic state feedback transformation that puts the system into a linear and decoupled form consisting of input-output chains of integrators. The algorithmic implementation of the latter idea will be shown in Sect. 3.3.

3.2 Control via approximate linearization

We now present a feedback controller for trajectory tracking based on standard linear control theory. The design makes use of the approximate linearization of the system equations about the desired trajectory, a procedure that leads to a time-varying system as seen in Sect. 2.2. A remarkable feature of this approach in the present case is the possibility of assigning a time-invariant eigenstructure to the closed-loop error dynamics.

In order to have a systematic procedure that can be easily extended to higher-dimensional wheeled robots (i.e., $n > 4$), the method is illustrated for the chained form case. However, similar design steps for a mobile robot in original coordinates can be found in [42].

For the chained-form representation (10), denote the desired state and input trajectory computed in correspondence to the reference cartesian trajectory as in Sect. 3.1 by $(x_{d1}(t), x_{d2}(t), x_{d3}(t), x_{d4}(t))$ and $u_d(t) = (u_{d1}(t), u_{d2}(t))$. Denote the state and input errors respectively as

$$\tilde{x}_i = x_{di} - x_i, \quad i = 1, \ldots, 4, \qquad \tilde{u}_j = u_{dj} - u_j, \quad j = 1, 2.$$

The nonlinear error equations are

$$\begin{aligned}
\dot{\tilde{x}}_1 &= \tilde{u}_1 \\
\dot{\tilde{x}}_2 &= \tilde{u}_2 \\
\dot{\tilde{x}}_3 &= x_{d2}u_{d1} - x_2 u_1 \\
\dot{\tilde{x}}_4 &= x_{d3}u_{d1} - x_3 u_1 .
\end{aligned}$$

Linearizing about the desired trajectory yields the following linear time-varying system

$$\dot{\tilde{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & u_{d1}(t) & 0 & 0 \\ 0 & 0 & u_{d1}(t) & 0 \end{bmatrix} \tilde{x} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ x_{d2}(t) & 0 \\ x_{d3}(t) & 0 \end{bmatrix} \tilde{u} = A(t)\tilde{x} + B(t)\tilde{u}.$$

This system shares the same controllability properties of eq. (6), which was obtained by linearizing the original robot equations (5) about the desired trajectory. For example, it is easily verified that the controllability rank condition is satisfied along a linear trajectory with constant velocity, which is obtained for $u_{d1}(t) \equiv u_{d1}$ (a constant nonzero value) and $u_{d2}(t) \equiv 0$, implying $x_{d2}(t) \equiv 0$ and $x_{d3}(t) \equiv x_{d3}(t_0)$.

Define the feedback term $\tilde{u}$ as the following linear time-varying law

$$\tilde{u}_1 = -k_1 \tilde{x}_1 \tag{28}$$

$$\tilde{u}_2 = -k_2 \tilde{x}_2 - \frac{k_3}{u_{d1}} \tilde{x}_3 - \frac{k_4}{u_{d1}^2} \tilde{x}_4, \tag{29}$$

with k_1 positive, and k_2, k_3, k_4 such that

$$\lambda^3 + k_2\lambda^2 + k_3\lambda + k_4$$

is a Hurwitz polynomial. With this choice, the closed-loop system matrix

$$A_{cl}(t) = \begin{bmatrix} -k_1 & 0 & 0 & 0 \\ 0 & -k_2 & -k_3/u_{d1}(t) & -k_4/u_{d1}^2(t) \\ -k_1 x_{d2}(t) & u_{d1}(t) & 0 & 0 \\ -k_1 x_{d3}(t) & 0 & u_{d1}(t) & 0 \end{bmatrix}$$

has constant eigenvalues with negative real part. In itself, this does not guarantee the asymptotic stability of de closed-loop time-varying system [20]. As a matter of fact, a general stability analysis for control law (28-29) is lacking. However, for specific choices of $u_{d1}(t)$ (bounded away from zero) and $u_{d2}(t)$, it is possible to use results on slowly-varying linear systems in order to prove asymptotic stability.

The location of the closed-loop eigenvalues in the open left half-plane may be chosen according to the general principle of obtaining fast convergence to zero of the tracking error with a reasonable control effort. For example, in order to assign two real negative eigenvalues in $-\lambda_1$ and $-\lambda_2$ and two complex eigenvalues with negative real part, modulus ω_n and damping coefficient ζ ($0 < \zeta \leq 1$), the gains k_i should be selected as

$$k_1 = \lambda_1, \quad k_2 = \lambda_2 + 2\zeta\omega_n, \quad k_3 = \omega_n^2 + 2\zeta\omega_n\lambda_2, \quad k_4 = \omega_n^2\lambda_2.$$

Note that the overall control input to the chained-form representation is

$$u = u_d + \tilde{u},$$

with a feedforward and a feedback component. In order to compute the actual input commands v for the car-like robot, one should use the input transformation (9). As a result, the driving and steering velocity inputs are expressed as nonlinear (and for v_2, also time-varying) feedback laws.

The choice (29) for the second control input requires $u_{d1} \neq 0$. Intuitively, placing the eigenvalues at a constant location will require larger gains as the desired motion of the variable x_1 is coming to a stop. One way to overcome this limitation is to assign the eigenvalues as functions of the input u_{d1}. For example, imposing (beside the eigenvalue in $-\lambda_1$) three coincident real eigenvalues in $-\alpha|u_{d1}|$, with $\alpha > 0$, we obtain

$$\tilde{u}_2 = -3\alpha|u_{d1}|\tilde{x}_2 - 3\alpha^2 u_{d1}\tilde{x}_3 - \alpha^3|u_{d1}|\tilde{x}_4, \tag{30}$$

in place of eq. (29). With this *input scaling* procedure, the second control input simply goes to zero when the desired trajectory x_{d1} stops. We point out that a rigorous Lyapunov-based proof can be derived for the asymptotic stability of the control law given by eqs. (28) and (30). This kind of procedure will be also used in Sect. 4.1.

Simulation results The simple controller (28-29) has been simulated for a car-like robot with $\ell = 1$ m tracking the sinusoidal trajectory (27), where $A = 1$ and $\omega = \pi$. The state at $t_0 = 0$ is

$$x_1(0) = -2, \quad x_2(0) = 0, \quad x_3(0) = A\,\omega, \quad x_4(0) = -1,$$

so that the car-like robot is initially off the desired trajectory. We have chosen $\lambda_1 = \lambda_2 = \omega_n = 5$ and $\zeta = 1$, resulting in four coincident closed-loop eigenvalues located at -5.

The obtained results are shown in Figs. 5–7 in terms of tracking errors on the original states x, y, θ and ϕ, and of actual control inputs v_1 and v_2 to the

car-like robot. Once convergence is achieved (approximately, after 2.5 sec), the control inputs virtually coincide with the feedforward commands associated to the nominal sinusoidal trajectory, as computed from eqs. (21) and (24).

Since the control design is based on approximate linearization, the controlled system is only locally asymptotically stable. However, extensive simulation shows that, also in view of the chained-form transformation, the region of asymptotic stability is quite large—although its accurate determination may be difficult. As a consequence, the car-like robot converges to the desired trajectory even for large initial errors. The transient behavior, however, may deteriorate in an unacceptable way.

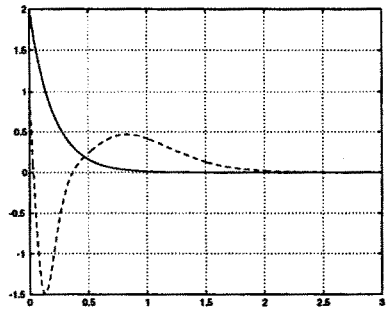


Fig. 5. Tracking a sinusoid with approximate linearization: x (—), y (– –) errors (m) vs. time (sec)

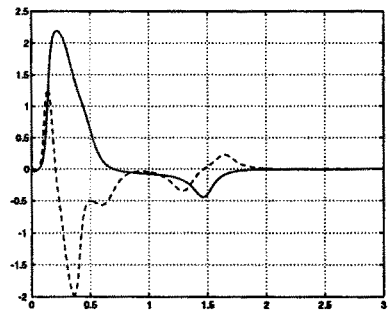


Fig. 6. Tracking a sinusoid with approximate linearization: θ (—), ϕ (– –) errors (rad) vs. time (sec)

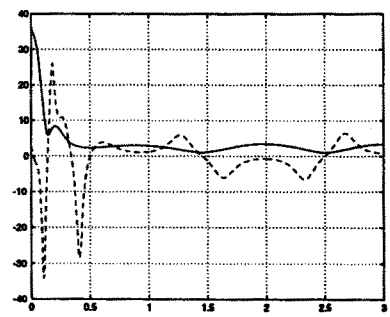


Fig. 7. Tracking a sinusoid with approximate linearization: v_1 (—) (m/sec), v_2 (– –) (rad/sec) vs. time (sec)

3.3 Control via exact feedback linearization

We now turn to the use of nonlinear feedback design for achieving global stabilization of the trajectory tracking error to zero.

It is well known in robotics that, if the number of generalized coordinates equals the number of input commands, one can use a nonlinear static (i.e., memoryless) state feedback law in order to transform exactly the nonlinear robot kinematics and/or dynamics into a linear system. In general, the linearity of the system equations is displayed only after a coordinate transformation in the state space. On the linear side of the problem, it is rather straightforward to complete the synthesis of a stabilizing controller. For example, this is the principle of the *computed torque* control method for articulated manipulators.

Actually, two types of exact linearization problems can be considered for a nonlinear system with outputs. Beside the possibility of transforming via feedback the whole set of differential equations into a linear system (*full-state linearization*), one may seek a weaker result in which only the input-output differential map is made linear (*input-output linearization*). Necessary and sufficient conditions exist for the solvability of both problems via static feedback, while only sufficient (but constructive) conditions can be given for the dynamic feedback case [18].

Consider a generic nonlinear system

$$\dot{x} = f(x) + G(x)u, \qquad z = h(x), \tag{31}$$

where x is the system state, u is the input, and z is the output to which we wish to assign a desired behavior (e.g., track a given trajectory). Assume the system is square, i.e., the number of inputs equals the number of outputs.

The input-output linearization problem via static feedback consists in looking for a control law of the form

$$u = a(x) + B(x)r, \tag{32}$$

with $B(x)$ nonsingular and r an external auxiliary input of the same dimension as u, in such a way that the input-output response of the closed-loop system (i.e., between the new inputs r and the outputs z) is linear. In the multi-input multi-output case, the solution to this problem automatically yields input-output decoupling, namely, each component of the output z will depend only on a single component of the input r.

In general, a nonlinear *internal dynamics* which does not affect the input-output behavior may be left in the closed-loop system. This internal dynamics reduces to the so-called *clamped dynamics* when the output z is constrained to follow a desired trajectory $z_d(t)$. In the absence of internal dynamics, full-state linearization is achieved. Conversely, when only input-output linearization is

obtained, the boundedness/stability of the internal dynamics should be analyzed in order to guarantee a feasible output tracking.

If static feedback does not allow to solve the problem, one can try to obtain the same results by means of a dynamic feedback compensator of the form

$$\begin{aligned} u &= a(x,\xi) + B(x,\xi)r \\ \dot{\xi} &= c(x,\xi) + D(x,\xi)r, \end{aligned} \tag{33}$$

where ξ is the compensator state of appropriate dimension. Again, the closed-loop system may or may not contain internal dynamics.

In its simplest form, which is suitable for the current application, the linearization algorithm proceeds by differentiating all system outputs until some of the inputs appear explicitly. At this point, one tries to invert the differential map in order to solve for the inputs. If the Jacobian of this map—referred to as the *decoupling matrix* of the system—is nonsingular, this procedure gives a static feedback law of the form (32) that solves the input-output linearization and decoupling problem.

If the decoupling matrix is singular, making it impossible to solve for all the inputs at the same time, one proceeds by adding integrators on a subset of the input channels. This operation, called *dynamic extension*, converts a system input into a state of a dynamic compensator, which is driven in turn by a new input. Differentiation of the outputs continues then until either it is possible to solve for the new inputs or the dynamic extension process has to be repeated. At the end, the number of added integrators will give the dimension of the state ξ of the nonlinear dynamic controller (33). The algorithm will terminate after a finite number of iterations if the system is *invertible* from the chosen outputs.

In any case, if the sum of the relative degrees (the order of differentiation of the outputs) equals the dimension of the (original or extended) state space, there is no internal dynamics and the same (static or dynamic, respectively) control law yields full-state linearization. In the following, we present both a static and a dynamic feedback controller for trajectory tracking.

Input-output linearization via static feedback For the car-like robot model (5), the natural output choice for the trajectory tracking task is

$$z = \begin{bmatrix} x \\ y \end{bmatrix}.$$

The linearization algorithm begins by computing

$$\dot{z} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix} v = A(\theta)v. \tag{34}$$

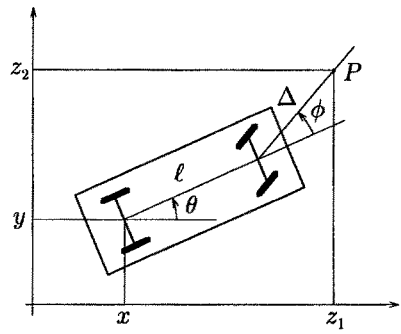


Fig. 8. Alternative output definition for a car-like robot

At least one input appears in both components of $\dot{z}$, so that $A(\theta)$ is the actual decoupling matrix of the system. Since this matrix is singular, static feedback fails to solve the input-output linearization and decoupling problem.

A possible way to circumvent this problem is to redefine the system output as

$$z = \begin{bmatrix} x + \ell\cos\theta + \Delta\cos(\theta+\phi) \\ y + \ell\sin\theta + \Delta\sin(\theta+\phi) \end{bmatrix}, \tag{35}$$

with $\Delta \neq 0$. This choice corresponds to selecting the representative point of the robot as P in Fig. 8, in place of the midpoint of the rear axle.

Differentiation of this new output gives

$$\dot{z} = \begin{bmatrix} \cos\theta - \tan\phi(\sin\theta + \Delta\sin(\theta+\phi)/\ell) & -\Delta\sin(\theta+\phi) \\ \sin\theta + \tan\phi(\cos\theta + \Delta\cos(\theta+\phi)/\ell) & \Delta\cos(\theta+\phi) \end{bmatrix} v = A(\theta,\phi)v.$$

Since $\det A(\theta,\phi) = \Delta/\cos\phi \neq 0$, we can set $\dot{z} = r$ (an auxiliary input value) and solve for the inputs v as

$$v = A^{-1}(\theta,\phi)r.$$

In the globally defined transformed coordinates (z_1, z_2, θ, ϕ), the closed-loop system becomes

$$\begin{aligned} \dot{z}_1 &= r_1 \\ \dot{z}_2 &= r_2 \\ \dot{\theta} &= \sin\phi\,[\cos(\theta+\phi)r_1 + \sin(\theta+\phi)r_2]\,/\ell \end{aligned} \tag{36}$$

$$\begin{aligned} \dot{\phi} = &-[\cos(\theta+\phi)\sin\phi/\ell + \sin(\theta+\phi)/\Delta]\,r_1 \\ &-[\sin(\theta+\phi)\sin\phi/\ell - \cos(\theta+\phi)/\Delta]\,r_2, \end{aligned} \tag{37}$$

which is input-output linear and decoupled (one integrator on each channel). We note that there exists a two-dimensional internal dynamics expressed by the differential equations for θ and ϕ.

In order to solve the trajectory tracking problem, we choose then

$$r_i = \dot{z}_{di} + k_{pi}(z_{di} - z_i), \qquad k_{pi} > 0, \qquad i = 1, 2, \tag{38}$$

obtaining exponential convergence of the output tracking error to zero, for any initial condition $(z_1(t_0), z_2(t_0), \theta(t_0), \phi(t_0))$. A series of remarks is now in order.

- While the two output variables converge to their reference trajectory with arbitrary exponential rate (depending on the choice of the k_{pi}'s in eq. (38)), the behavior of the variables θ and ϕ cannot be specified at will because it follows from the last two equations of (36).
- A complete analysis would require the study of the stability of the time-varying closed-loop system (36), with r given by eq. (38). In practice, one is interested in the boundedness of θ and ϕ along the nominal output trajectory. This study may not be trivial for higher-dimensional wheeled mobile robots, where the internal dynamics has dimension $n - 2$.
- Having redefined the system outputs as in eq. (35), one has two options for generating the reference output trajectory. The simplest choice is to directly plan a cartesian motion to be executed by the point P. On the other hand, if the planner generates a desired motion $x_d(t), y_d(t)$ for the rear axle midpoint (with associated $v_{d1}(t), v_{d2}(t)$ computed from eqs. (21) and (24)), this must be converted into a reference motion for P by forward integration of the car-like equations, with $v = v_d(t)$ and use of the output equation (35). In both cases, there is no smoothness requirement for $z_d(t)$ which may contain also discontinuities in the path tangent.
- The output choice (35) is not the only one leading to input-output linearization and decoupling by static feedback. As a matter of fact, the first two variables of the chained-form transformation (8) are another example of linearizing outputs, with static feedback given by (9).

Full-state linearization via dynamic feedback In order to design a tracking controller directly for the cartesian outputs (x, y) of the car-like robot, dynamic extension is required in order to overcome the singularity of the decoupling matrix in eq. (34). Although the linearization procedure can be continued using the original kinematic description (5), we will apply it here to the chained-form representation (10) as a first step toward the extension to higher-dimensional systems.

In accordance with the task definition, choose the two system outputs as

$$z = \begin{bmatrix} x_1 \\ x_4 \end{bmatrix},$$

namely the x and y coordinates of the robot. Differentiating z with respect to time gives

$$\dot{z} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ x_3 & 0 \end{bmatrix} u,$$

where the input u_2 does not appear, so that the decoupling matrix is singular. In order to proceed with the differentiation, an integrator (with state denoted by ξ_1) is added on the first input

$$u_1 = \xi_1, \qquad \dot{\xi}_1 = u_1', \tag{39}$$

with u_1' a new auxiliary input. Using eq. (39), we can rewrite the first derivative of the output as

$$\dot{z} = \begin{bmatrix} \xi_1 \\ \xi_1 x_3 \end{bmatrix},$$

which is independent from the inputs u_1' and u_2 of the extended system. In this way, differentiation of the original input signal at the next step of the procedure is avoided. We have

$$\ddot{z} = \begin{bmatrix} \dot{\xi}_1 \\ \dot{x}_3\xi_1 + x_3\dot{\xi}_1 \end{bmatrix} = \begin{bmatrix} 0 \\ x_2\xi_1^2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ x_3 & 0 \end{bmatrix} \begin{bmatrix} u_1' \\ u_2 \end{bmatrix}.$$

As u_2 does not appear yet, we add another integrator (with state denoted by ξ_2) on the input u_1'

$$u_1' = \xi_2, \qquad \dot{\xi}_2 = u_1'', \tag{40}$$

obtaining

$$\ddot{z} = \begin{bmatrix} \dot{\xi}_2 \\ x_2\xi_1^2 + x_3\xi_2 \end{bmatrix}.$$

Finally, the last differentiation gives

$$\dddot{z} = \begin{bmatrix} 0 \\ 3x_2\xi_1\xi_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ x_3 & \xi_1^2 \end{bmatrix} \begin{bmatrix} u_1'' \\ u_2 \end{bmatrix}. \tag{41}$$

The matrix weighting the inputs is nonsingular provided that $\xi_1 \neq 0$. Under such assumption—on which we will come back later—we set $\dddot{z} = r$ (an auxiliary input value) and solve eq. (41) for

$$\begin{bmatrix} u_1'' \\ u_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ (r_2 - x_3 r_1 - 3x_2\xi_1\xi_2)/\xi_1^2 \end{bmatrix}. \tag{42}$$

Putting together the dynamic extensions (39) and (40) with eq. (42), the resulting nonlinear dynamic feedback controller

$$\begin{aligned} u_1 &= \xi_1 \\ u_2 &= (r_2 - x_3 r_1 - 3x_2\xi_1\xi_2)/\xi_1^2 \\ \dot{\xi}_1 &= \xi_2 \\ \dot{\xi}_2 &= r_1 \end{aligned} \tag{43}$$

transforms the original system into two decoupled chains of three input-output integrators

$$\begin{aligned} \dddot{z}_1 &= r_1 \\ \dddot{z}_2 &= r_2. \end{aligned}$$

The original system in chained form had four states, whereas the dynamic controller has two additional states. All these six states are found in the above input-output description, and hence there is no internal dynamics left. Thus, full-state linearization has been obtained.

On the linear and decoupled system, it is easy to complete the control design with a globally stabilizing feedback for the desired trajectory (independently on each integrator chain). To this end, let

$$r_i = \dddot{z}_{di} + k_{ai}(\ddot{z}_{di} - \ddot{z}_i) + k_{vi}(\dot{z}_{di} - \dot{z}_i) + k_{pi}(z_{di} - z_i), \qquad i = 1, 2, \tag{44}$$

where the feedback gains are such that the polynomials

$$\lambda^3 + k_{ai}\lambda^2 + k_{vi}\lambda + k_{pi}, \qquad i = 1, 2,$$

are Hurwitz, and z, $\dot{z}$, $\ddot{z}$ are computed from the intermediate steps of the dynamic extension algorithm as

$$\begin{aligned} z &= \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} \\ \dot{z} &= \begin{bmatrix} \xi_1 \\ x_3\xi_1 \end{bmatrix} \\ \ddot{z} &= \begin{bmatrix} \xi_2 \\ x_2\xi_1^2 + x_3\xi_2 \end{bmatrix}. \end{aligned} \tag{45}$$

In order to initialize the chained-form system and the associated dynamic controller for exact reproduction of the desired output trajectory, we can set

$z = z_d(t)$ and solve eqs. (45) at time $t = t_0$:

$$\begin{aligned}
x_1(t_0) &= z_{d1}(t_0) \quad (= x_d(t_0)) \\
x_2(t_0) &= [\dot{z}_{d1}(t_0)\ddot{z}_{d2}(t_0) - \ddot{z}_{d1}(t_0)\dot{z}_{d2}(t_0)] \,/\, \dot{z}_{d1}^3(t_0) \\
x_3(t_0) &= \dot{z}_{d2}(t_0)/\dot{z}_{d1}(t_0) \\
x_4(t_0) &= z_{d2}(t_0) \quad (= y_d(t_0)) \\
\xi_1(t_0) &= \dot{z}_{d1}(t_0) \\
\xi_2(t_0) &= \ddot{z}_{d1}(t_0).
\end{aligned}$$

Any other initialization of the robot and/or the dynamic controller will produce a transient state error that converges exponentially to zero, with the rate specified by the chosen gains in eq. (44).

As mentioned in Sect. 3.1, only trajectories $z_d(t) = (x_d(t), y_d(t))$ with continuous second time derivatives are exactly reproducible. In the presence of lesser smoothness, the car-like robot will deviate from the desired trajectory. Nonetheless, after the occurrence of isolated discontinuities, the feedback controller (43–44) will be able to drive the vehicle back to the remaining part of the smooth trajectory at an exponential rate.

The above approach can be easily extended to the general case of the $(2, n)$ chained form (7). In fact, such representation can be fully transformed via dynamic feedback into decoupled strings of input-output integrators by defining the system output as (x_1, x_n). This result is summarized in the following proposition.

Proposition 3.1. *Consider the* $(2, n)$ *chained-form system (7) and define its output as*

$$z = \begin{bmatrix} x_1 \\ x_n \end{bmatrix}. \tag{46}$$

By using a nonlinear dynamic feedback controller of dimension $n-2$*, the system can be fully transformed into a linear one consisting of two decoupled chains of* $n-1$ *integrators, provided that* $u_1 \neq 0$.

Proof We will provide a constructive solution. Let the dynamic extension be composed of $n-2$ integrators added on input u_1

$$u_1^{(n-2)} = \bar{u}_1, \tag{47}$$

with the input u_2 unchanged. Denote the states of these integrators by $\xi_1, \ldots, \xi_{n-2}$, so that a state-space representation of eq. (47) is

$$\begin{aligned} u_1 &= \xi_1 \\ \dot{\xi}_1 &= \xi_2 \\ \dot{\xi}_2 &= \xi_3 \\ &\vdots \\ \dot{\xi}_{n-2} &= \bar{u}_1. \end{aligned} \tag{48}$$

The extended system consisting of eqs. (7) and (48) is

$$\begin{aligned} \dot{x}_1 &= \xi_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_2\xi_1 \\ \dot{x}_4 &= x_3\xi_1 \\ &\vdots \\ \dot{x}_{n-1} &= x_{n-2}\xi_1 \\ \dot{x}_n &= x_{n-1}\xi_1 \\ \dot{\xi}_1 &= \xi_2 \\ \dot{\xi}_2 &= \xi_3 \\ &\vdots \\ \dot{\xi}_{n-3} &= \xi_{n-2} \\ \dot{\xi}_{n-2} &= \bar{u}_1. \end{aligned} \tag{49}$$

By applying the linearization algorithm, we have for the first few derivatives of the output (46):

$$\begin{aligned} \dot{z} &= \begin{bmatrix} \xi_1 \\ x_{n-1}\xi_1 \end{bmatrix} \\ \ddot{z} &= \begin{bmatrix} \xi_2 \\ x_{n-2}\xi_1^2 + x_{n-1}\xi_2 \end{bmatrix} \\ \dddot{z} &= \begin{bmatrix} \xi_3 \\ x_{n-3}\xi_1^3 + x_{n-1}\xi_3 + 3\xi_1\xi_2 x_{n-2} \end{bmatrix} \\ z^{(4)} &= \begin{bmatrix} \xi_4 \\ x_{n-4}\xi_1^4 + x_{n-1}\xi_4 + (6\xi_1^2\xi_2 x_{n-3} + 4\xi_1\xi_3 x_{n-2} + 3\xi_2^2 x_{n-2}) \end{bmatrix}, \\ &\vdots \end{aligned}$$

so that the structure of the $(n-2)$-th derivative is

$$z^{(n-2)} = \begin{bmatrix} \xi_{n-2} \\ x_2\xi_1^{n-2} + x_{n-1}\xi_{n-2} + f(\xi_1, \xi_2, \ldots, \xi_{n-3}, x_3, x_4, \ldots, x_{n-2}) \end{bmatrix},$$

where f is a polynomial function of its arguments. The expressions of the output (46) together with its derivatives up to the $(n-2)$-th order induce a diffeomorphism between $(x_1, \ldots, x_n, \xi_1, \ldots, \xi_{n-2})$ and $(z, \dot{z}, \ldots, z^{(n-2)})$, which is globally defined except for the manifold $\xi_1 = 0$.

We obtain finally

$$z^{(n-1)} = \begin{bmatrix} 0 \\ g(\xi_1, \xi_2, \ldots, \xi_{n-2}, x_2, x_3, \ldots, x_{n-2}) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ x_{n-1} & \xi_1^{n-2} \end{bmatrix} \begin{bmatrix} \bar{u}_1 \\ u_2 \end{bmatrix}, \quad (50)$$

where g is a polynomial function of its arguments. The decoupling matrix of the extended system is nonsingular provided that $\xi_1 \neq 0$ or, equivalently, that $u_1 \neq 0$. Under this assumption, we can set $z^{(n-1)} = r$ and solve eq. (50) for $(\bar{u}_1, u_2)$. Reorganizing with eq. (48), we conclude that the following nonlinear dynamic controller of dimension $n-2$

$$\begin{aligned} u_1 &= \xi_1 \\ u_2 &= \left[r_2 - x_{n-1} r_1 - g(\xi_1, \xi_2, \ldots, \xi_{n-2}, x_2, x_3, \ldots, x_{n-2})\right] / \xi_1^{n-2} \\ \dot{\xi}_1 &= \xi_2 \\ &\vdots \\ \dot{\xi}_{n-3} &= \xi_{n-2} \\ \dot{\xi}_{n-2} &= r_1 \end{aligned} \quad (51)$$

transforms the original chained-form system (7) with output (46) into the input-output linear and decoupled system

$$z^{(n-1)} = \begin{bmatrix} x_1^{(n-1)} \\ x_n^{(n-1)} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = r.$$

Since the number of the input-output integrators $(2(n-1))$ equals the number of states of the extended system $(n + (n-2))$, there is no internal dynamics in the closed-loop system and thus we have obtained full-state linearization and input-output decoupling. ∎

The above result indicates that dynamic feedback linearization offers a viable control design tool for trajectory tracking, even for higher-dimensional kinematic models of wheeled mobile robots (e.g., the N-trailer system). The same dynamic extension technique can be directly applied to the original kinematic equations of the wheeled mobile robot, without resorting to the chained-form transformation. In particular, for the car-like robot (5), similar computa-

tions show that the dynamic controller takes the form:

$$\begin{aligned} v_1 &= \xi_1 \\ v_2 &= -3\xi_2 \cos^2\phi \tan\phi/\xi_1 - \ell r_1 \cos^2\phi \sin\theta/\xi_1^2 + \ell r_2 \cos^2\phi \cos\theta/\xi_1^2 \\ \dot{\xi}_1 &= \xi_2 \\ \dot{\xi}_2 &= \xi_1^3 \tan^2\phi/\ell^2 + r_1 \cos\theta + r_2 \sin\theta. \end{aligned} \tag{52}$$

The external inputs r_1 and r_2 are chosen as in (44), with the values of z, $\dot{z}$ and $\ddot{z}$ given by

$$\begin{aligned} z &= \begin{bmatrix} x \\ y \end{bmatrix} \\ \dot{z} &= \begin{bmatrix} \xi_1 \cos\theta \\ \xi_1 \sin\theta \end{bmatrix} \\ \ddot{z} &= \begin{bmatrix} -\xi_1^2 \tan\phi \sin\theta/\ell + \xi_2 \cos\theta \\ \xi_1^2 \tan\phi \cos\theta/\ell + \xi_2 \sin\theta \end{bmatrix}. \end{aligned} \tag{53}$$

The derivation of the initial conditions on (x, y, θ, ϕ) and (ξ_1, ξ_2) allowing for exact reproduction of a smooth trajectory is straightforward using eqs. (53).

The main limitation of the dynamic feedback linearization approach is the requirement that the compensator state variable ξ_1 (which corresponds to v_1 if linearization is performed on the original car-like equations, or to u_1 if it is performed on its chained-form representation) should never be zero. In fact, in this case the second control input (i.e., v_2 in eq. (52) and u_2 in eq. (43) or, more in general, in eq. (51)) could diverge. It has been shown that the occurrence of this singularity in the dynamic extension process is structural for nonholonomic systems [14]. Therefore, this approach as such is feasible only for trajectory tracking.

In addition, we note the following facts with specific reference to the controller (52–53) for the car-like robot.

- If the desired trajectory is smooth and persistent, the nominal control input v_{d1} does not decay to zero. As the robot is guaranteed to converge exponentially to the desired trajectory, also the actual command v_1 will eventually be bounded away from zero. On the other hand, exact reproduction of trajectories with linear velocity vanishing to zero (e.g., trajectories with cusps, where the robot should stop and reverse the direction of motion) is not allowed with this control scheme.
- Even for smooth persistent trajectories, problems may arise if the command v_1 crosses zero during an initial transient. However, this situation can be avoided by suitably choosing the initialization of the dynamic controller (i.e., the states ξ_1 and ξ_2), which is in practice an additional degree of

freedom in the design. As a matter of fact, a simple way to keep the actual commands bounded is to reset the state ξ_1 whenever its value falls below a given threshold. Note that this would result in an input command v that is discontinuous with respect to time.

The problem of tracking a trajectory starting (or ending) with zero linear velocity using the above approach can be solved by separating geometric from timing information in the control law, along the same lines indicated in Sect. 3.1. Suppose that a smooth path of finite length L has to be tracked starting and ending with zero velocity, and let σ be the path parameter. The timing law $\sigma(t)$ can be any increasing function such that

$$\sigma(0) = 0, \quad \sigma(T) = L, \quad \dot{\sigma}(0) = \dot{\sigma}(T) = 0,$$

where T is the final time at which the motion ends. The car-like equations can be rewritten in the path parameter domain as

$$\begin{aligned} x' &= \cos\theta\, w_1 \\ y' &= \sin\theta\, w_1 \\ \theta' &= \tan\phi\, w_1/\ell \\ \phi' &= w_2, \end{aligned} \tag{54}$$

with the actual velocity commands v_i related to the new inputs w_i by

$$v_i(t) = w_i(\sigma(t))\dot{\sigma}(t), \qquad i = 1, 2.$$

For system (54), one can design a dynamic feedback achieving full-state linearization as before. In this case, tracking errors will converge exponentially to zero in the σ-domain (instead of the t-domain). Moreover, the control law is always well-defined since it is possible to show that in the denominator of w_2 only the linear pseudovelocity w_1 appears, a geometric quantity which is always nonzero being related to the path tangent.

Simulation results In order to compare the performance of linear and nonlinear control design, the nonlinear dynamic controller (43-44) computed for the chained-form representation has been used to track the same sinusoidal trajectory (27) of Sect. 3.2. The initial condition at $t_0 = 0$ of the car-like robot (of length $\ell = 1$ m) is the same as before (off the trajectory)

$$x_1(0) = -2, \quad x_2(0) = 0, \quad x_3(0) = A\omega, \quad x_4(0) = -1,$$

with the initial state of the dynamic compensator set at

$$\xi_1(0) = 1, \quad \xi_2(0) = 0.$$

As for the stabilizing part of the controller, we have chosen the same gains for both input-output channels, with three coincident closed-loop eigenvalues located at -5 ($k_{ai} = 15$, $k_{vi} = 75$, $k_{pi} = 125$, $i = 1, 2$).

The results are shown in Figs. 9–11, again in terms of errors on the original states x, y, θ and ϕ, and of the actual control inputs v_1, v_2 for the car-like robot. A comparison with the analogous plots in Figs. 5–7 shows that the peaks of the transient errors are approximately halved in this particular case. Also, the control effort on the linear velocity v_1 in Fig. 11 does not reach the large initial value of Fig. 7, while the control input v_2 has a lower average value. After achieving convergence, the input commands of the nonlinear dynamic controller are identical to those of the linear one, for they both reduce to the nominal feedforward needed to execute the desired trajectory. Moreover, as a result of the imposed linear dynamics of the feedback controlled system, the transient behavior of the errors is globally exponentially converging to zero, i.e., for any initial conditions of the car-like robot and of the dynamic compensator.

We have also simulated the dynamic feedback controller (52–53) designed directly on the car-like model. Results for a circular and an eight-shaped trajectory are reported in Figs. 12–19, assuming a length $\ell = 0.1$ m for the car. Note that the small peak of the x error in Fig. 13 is only due to an initial mismatch of the robot state with respect to the value specified by the higher-order derivatives of $x_d(t)$ (in particular, of $\dot{x}_d(0)$). In fact, in view of the decoupling property induced by the controller, the value of the initial error along each cartesian direction does not affect the error behavior and the control action in the other direction.

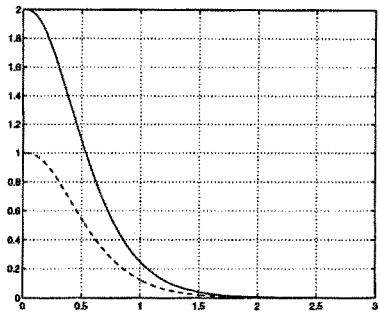


Fig. 9. Tracking a sinusoid with dynamic feedback linearization: x (—), y (– –) errors (m) vs. time (sec)

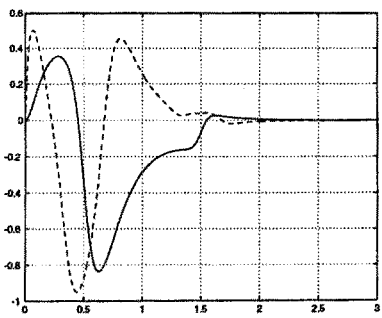


Fig. 10. Tracking a sinusoid with dynamic feedback linearization: θ (—), ϕ (– –) errors (rad) vs. time (sec)

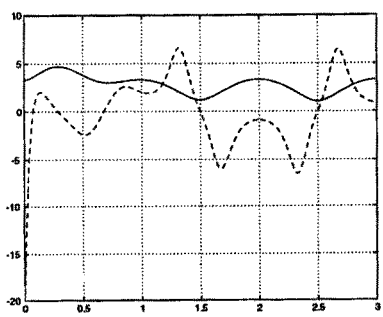


Fig. 11. Tracking a sinusoid with dynamic feedback linearization: v_1 (—) (m/sec), v_2 (– –) (rad/sec) vs. time (sec)

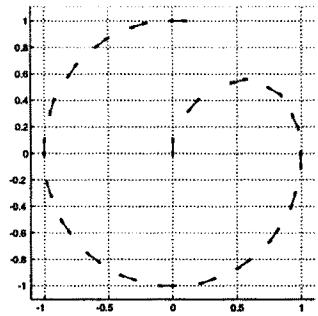


Fig. 12. Tracking a circle with dynamic feedback linearization: stroboscopic view of the cartesian motion

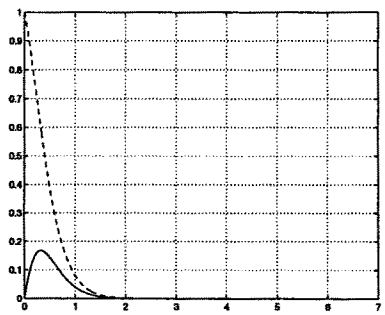


Fig. 13. Tracking a circle with dynamic feedback linearization: x (—), y (– –) errors (m) vs. time (sec)

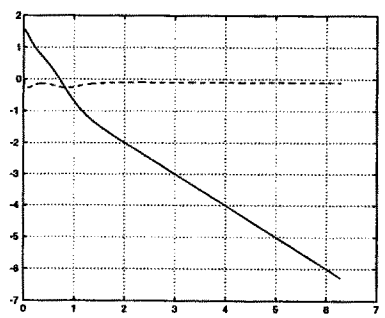


Fig. 14. Tracking a circle with dynamic feedback linearization: θ (—), ϕ (– –) (rad) vs. time (sec)

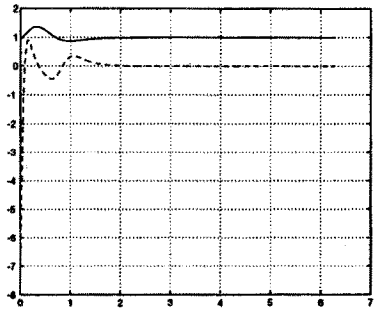


Fig. 15. Tracking a circle with dynamic feedback linearization: v_1 (—) (m/sec), v_2 (– –) (rad/sec) vs. time (sec)

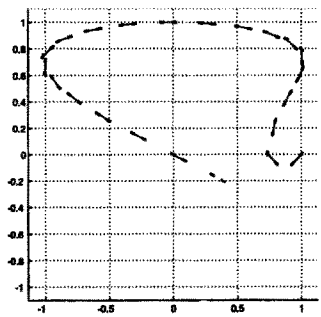


Fig. 16. Tracking an eight figure with dynamic feedback linearization: stroboscopic view of the cartesian motion

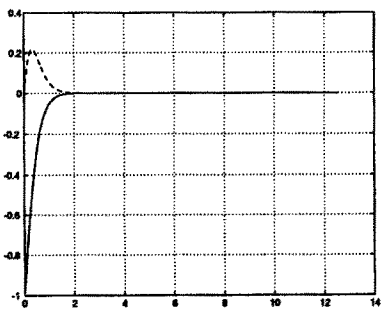


Fig. 17. Tracking an eight figure with dynamic feedback linearization: x (—), y (– –) errors (m) vs. time (sec)

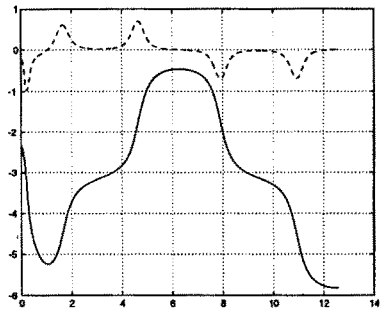


Fig. 18. Tracking an eight figure with a dynamic feedback linearization: θ (—), ϕ (– –) (rad) vs. time (sec)

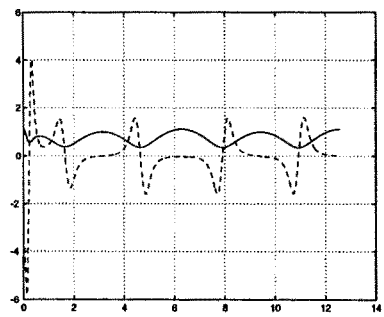


Fig. 19. Tracking an eight figure with a dynamic feedback linearization: v_1 (—) (m/sec), v_2 (– –) (rad/sec) vs. time (sec)

4 Path following and point stabilization

In this section, we address the problem of driving the car-like robot to a desired, fixed configuration by using only the current error information, without the need of planning a trajectory joining the initial point to the final destination. In doing this, a control solution for the path following task is obtained as an intermediate step.

As shown in Sect. 2, for nonholonomic mobile robots the point stabilization problem is considerably more difficult than trajectory tracking and path

following. In particular, smooth pure state feedback does not solve the problem. Recently, the idea of including an exogenous time-varying signal in the controller has proven to be successful for achieving asymptotic stability [38]. Roughly speaking, the underlying logic is that such a signal allows all components of the configuration error to be reflected on the control inputs, so that the error itself can be asymptotically reduced to zero.

Some physical insight on the role of time-varying feedback can be gained by thinking about a parallel parking maneuver of a real car. We can reasonably assume that a human driver controls the vehicle through a front wheel steering command and a linear velocity command. In order to bring to zero the error in the lateral direction—along which the car cannot move directly—experience indicates that an approximately periodic forward/backward motion should be imposed to the car. This motion is somewhat independent from the lateral position error with respect to the goal and is used only to sustain the generation of a net side motion through the combined action of the steering command, which is instead a function of the position error.

Although natural, this strategy is hard to generalize to more complex vehicle kinematics, whose maneuverability is less intuitive. For this reason, we have chosen to present here two methods of wide applicability that are designed on the chained-form representation of the system. In order to emphasize the generality of the controller design, the case of an n-dimensional system is considered. We give, however, some details for the case $n = 4$.

Both the presented feedback laws are time-varying, but they differ in their dependence from the current state as well as in some methodological aspects. The first control is either smooth or at least continuous with respect to the robot state. The second is nonsmooth, in the sense that the state is measured and fed back only at discrete instants of time. Nevertheless, discarding this time-discretization aspect, it is also basically continuous with respect to the state at the desired configuration, contrary to other time-invariant discontinuous feedback laws (see Sect. 6). Both controllers provide inputs that are continuous with respect to time and are globally defined on the chained-form representation.

Throughout this section, it will be assumed without loss of generality that the desired configuration coincides with the origin of the state space.

4.1 Control via smooth time-varying feedback

The smooth feedback stabilization method presented here was proposed in [44]. It exploits the internal structure of chained systems in order to decompose the solution approach in two design phases. In the first phase, it is assumed that one control input is given and satisfies some general requirements. The other control is then designed for achieving stabilization of an $(n-1)$-dimensional

subvector of the system state. At this stage, a solution to the path following problem has already been obtained. In the second phase, the design is completed by specifying the first control so as to guarantee convergence of the remaining variable while preserving the overall closed-loop stability.

As a preliminary step, reorder for convenience the variables of the chained form by letting

$$\mathcal{X} = (\chi_1, \chi_2, \ldots, \chi_{n-1}, \chi_n) = (x_1, x_n, \ldots, x_3, x_2).$$

As a consequence, the chained form system (7) becomes

$$\begin{aligned} \dot{\chi}_1 &= u_1 \\ \dot{\chi}_2 &= \chi_3 u_1 \\ \dot{\chi}_3 &= \chi_4 u_1 \\ &\vdots \\ \dot{\chi}_{n-1} &= \chi_n u_1 \\ \dot{\chi}_n &= u_2, \end{aligned} \tag{55}$$

or equivalently

$$\dot{\mathcal{X}} = h_1(\mathcal{X})u_1 + h_2 u_2, \qquad h_1(\mathcal{X}) = \begin{bmatrix} 1 \\ \chi_3 \\ \chi_4 \\ \vdots \\ \chi_n \\ 0 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{56}$$

For the car-like robot, the above reordering is simply an exchange between the second and fourth coordinates. Therefore, the cartesian position of the rear wheel is (χ_1, χ_2). Analogously, for an N-trailer robot, χ_1 and χ_2 represent the cartesian coordinates of the midpoint of the last trailer axle.

Let $\mathcal{X} = (\chi_1, \mathcal{X}_2)$, with $\mathcal{X}_2 = (\chi_2, \chi_3, \ldots, \chi_n)$. In the following, we shall first pursue the stabilization of $\mathcal{X}_2$ to zero, and then the complete stabilization of $\mathcal{X}$ to the origin.

Path following via input scaling When u_1 is assigned as a function of time, the chained system (56) can be written as

$$\begin{aligned}\dot{\tilde{\chi}}_1 &= 0 \\ \dot{\mathcal{X}}_2 &= \begin{bmatrix} 0 & u_1(t) & 0 & \cdots & \cdots & 0 \\ 0 & 0 & u_1(t) & 0 & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & \cdots & \cdots & \cdots & u_1(t) & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & u_1(t) \\ 0 & 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \mathcal{X}_2 + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2,\end{aligned} \tag{57}$$

having set

$$\tilde{\chi}_1 = \chi_1 - \int_0^t u_1(\tau)d\tau.$$

The first equation in (57) clearly shows that, when the input u_1 is a priori assigned, the system is no more controllable. However, the structure of the differential equations for $\mathcal{X}_2$ is reminiscent of the controllable canonical form for linear systems. In particular, when u_1 is constant and nonzero, system (57) becomes time-invariant and its second part is clearly controllable. As a matter of fact, controllability holds whenever $u_1(t)$ is a piecewise-continuous, bounded, and strictly positive (or negative) function. Under these assumptions, x_1 varies monotonically with time and differentiation with respect to time can be replaced by differentiation with respect to χ_1, being

$$\frac{d}{dt} = \frac{d}{d\chi_1}\dot{\chi}_1 = \frac{d}{d\chi_1}u_1,$$

and thus

$$\text{sign}(u_1)\frac{d}{d\chi_1} = \frac{1}{|u_1|} \cdot \frac{d}{dt}.$$

This change of variable is equivalent to an input scaling procedure (see Sect. 3.2). Then, the second part of the system may be rewritten as

$$\begin{aligned} \chi_2^{[1]} &= \text{sign}(u_1)\chi_3 \\ \chi_3^{[1]} &= \text{sign}(u_1)\chi_4 \\ &\vdots \\ \chi_{n-1}^{[1]} &= \text{sign}(u_1)\chi_n \\ \chi_n^{[1]} &= \text{sign}(u_1)u_2', \end{aligned} \tag{58}$$

with the definitions

$$\chi_i^{[j]} = \text{sign}(u_1)\frac{d^j\chi_i}{d\chi_1^j} \quad \text{and} \quad u_2' = \frac{u_2}{u_1}.$$

Equation (58) is a linear time-invariant system, an equivalent input-output representation of which is

$$\chi_2^{[n-1]} = \text{sign}(u_1)^{n-1}u_2'.$$

Such system is controllable and admits an exponentially stable linear feedback in the form

$$u_2'(\mathcal{X}_2) = -\text{sign}(u_1)^{n-1}\sum_{i=1}^{n-1} k_i\chi_2^{[i-1]}, \tag{59}$$

where the gains $k_i > 0$ are chosen so as to satisfy the Hurwitz stability criterion. Hence, the time-varying control

$$u_2(\mathcal{X}_2, t) = u_1(t)u_2'(\mathcal{X}_2) \tag{60}$$

globally asymptotically stabilizes the origin $\mathcal{X}_2 = 0$.

The above approach provides a solution to the path following problem. Consider in particular the case of a car-like robot. We have seen at the end of Sect. 2.3 that the system equations (14) in path coordinates can be transformed in chained form. By reordering the variables as in eq. (55), χ_1 represents the arc length s along the path, χ_2 is the distance d between the car and the path, while χ_3 and χ_4 are related to the car steering angle ϕ and to the relative orientation θ_p between the path and the car. Path following requires zeroing the χ_2, χ_3 and χ_4 variables (i.e., $\mathcal{X}_2 = 0$), independently from χ_1. Then, for any piecewise-continuous, bounded, and strictly positive (or negative) u_1, eq. (59) is particularized as

$$u_2'(\chi_2, \chi_3, \chi_4) = -\text{sign}(u_1)[k_1\chi_2 + k_2\,\text{sign}(u_1)\chi_3 + k_3\chi_4].$$

Using eq. (60), the path following feedback control law is

$$u_2(\chi_2, \chi_3, \chi_4, t) = -k_1|u_1(t)|\chi_2 - k_2u_1(t)\chi_3 - k_3|u_1(t)|\chi_4,$$

which can be compared with eq. (30) to appreciate the analogy. Such an approach was originally proposed in [42] for the path following of a unicycle. From the above developments, it is clear that it can be applied to any mobile robot which can be converted into chained form.

Skew-symmetric chained forms and Lyapunov control design We show now, by introducing a modified chained form, that it is possible to stabilize globally the origin $\mathcal{X}_2 = 0$ under more general hypotheses, namely that $|u_1(t)|$, $|\dot{u}_1(t)|$ are bounded and $u_1(t)$ does not asymptotically tend to zero. An important difference with respect to the previous analysis is that $u_1(t)$ is allowed to pass through zero. From there, it will be relatively simple to derive a class of smooth time-varying feedback laws which stabilize globally the origin $\mathcal{X} = 0$ of the complete system (point stabilization).

Consider the following change of coordinates

$$\begin{aligned} z_1 &= \chi_1 \\ z_2 &= \chi_2 \\ z_3 &= \chi_3 \\ z_{j+3} &= k_j z_{j+1} + L_{h_1} z_{j+2}, \qquad j = 1, \dots, n-3, \end{aligned} \tag{61}$$

where k_j $(j = 1, \dots, n-3)$ is a real positive number and $L_{h_1} z_j = \frac{\partial z_j}{\partial \mathcal{X}} h_1(\mathcal{X})$ is the Lie derivative of z_j along the vector field h_1. One easily verifies that eq. (61) is a linear, invertible change of coordinates, since the associated Jacobian matrix is of full rank. In particular, $\mathcal{X} = 0$ and $\mathcal{X}_2 = 0$ are respectively equivalent to $Z = 0$ and $Z_2 = 0$, having set $Z = (z_1, Z_2)$, $Z_2 = (z_2, z_3, \dots, z_n)$. Moreover, it is $L_{h_2} z_i = 0$ $(i = 1, \dots, n-1)$ and $L_{h_2} z_n = 1$.

Taking the time derivative of z_{j+3} and using eq. (56) gives

$$\dot{z}_{j+3} = \frac{\partial z_{j+3}}{\partial \mathcal{X}} \dot{\mathcal{X}} = (L_{h_1} z_{j+3}) u_1 + (L_{h_2} z_{j+3}) u_2,$$

and from eq. (61)

$$L_{h_1} z_{j+3} = -k_{j+1} z_{j+2} + z_{j+4}.$$

As a result, we obtain

$$\dot{z}_{j+3} = (-k_{j+1} z_{j+2} + z_{j+4}) u_1 \qquad j = 0, \dots, n-4$$

and for the last differential equation

$$\dot{z}_n = L_{h_1} z_n u_1 + u_2.$$

The original chained system (55) has thus been converted into the following *skew-symmetric* chained form

$$\begin{aligned} \dot{z}_1 &= u_1 \\ \dot{z}_2 &= u_1 z_3 \\ \dot{z}_{j+3} &= -k_{j+1} u_1 z_{j+2} + u_1 z_{j+4}, \qquad j = 0, \dots, n-4, \\ \dot{z}_n &= -k_{n-2} u_1 z_{n-1} + w_2, \end{aligned} \tag{62}$$

where it was convenient to define the new input signal

$$w_2 = (k_{n-2} z_{n-1} + L_{h_1} z_n) u_1 + u_2. \tag{63}$$

The skew-symmetric structure of the above form is clear when writing the system as follows

$$\begin{aligned} \dot{z}_1 &= u_1 \\ \dot{Z}_2 &= \operatorname{diag}\{1, k_1, k_1 k_2, \ldots, \prod_{j=1}^{n-2} k_j\} \cdot (S(u_1) Z_2 + b\, w_2), \end{aligned}$$

where

$$S(u_1) = \begin{bmatrix} 0 & u_1 & & & & & \\ -u_1 & 0 & \frac{u_1}{k_1} & & & & \\ & -\frac{u_1}{k_1} & 0 & \frac{u_1}{k_1 k_2} & & & \\ & & -\frac{u_1}{k_1 k_2} & 0 & & & \\ & & & \cdots & & & \\ & & & & 0 & \frac{u_1}{\prod_{j=1}^{n-3} k_j} & \\ & & & & -\frac{u_1}{\prod_{j=1}^{n-3} k_j} & 0 & \end{bmatrix}, \qquad b = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ \frac{1}{\prod_{j=1}^{n-2} k_j} \end{bmatrix}.$$

The interest of the skew-symmetric form is that it is naturally suited for a Lyapunov-like analysis, as illustrated by the following result.

Proposition 4.1. *Assume that* $|u_1(t)|$ *and* $|\dot{u}_1(t)|$ *are bounded, and let*

$$w_2 = -k_{w_2}(u_1) z_n, \tag{64}$$

where $k_{w_2}(\cdot)$ *is a continuous application strictly positive on* $\mathbb{R} - \{0\}$. *If this control law is applied to system (62), the positive function*

$$V(Z_2) = \frac{1}{2}\left(z_2^2 + \frac{1}{k_1} z_3^2 + \frac{1}{k_1 k_2} z_4^2 + \cdots + \frac{1}{\prod_{j=1}^{n-2} k_j} z_n^2\right)$$

is nonincreasing along the closed-loop system solutions and asymptotically converges to a limit value $V_{\lim}$ *which depends on the initial conditions. Moreover, if* $u_1(t)$ *does not asymptotically tend to zero, it is* $V_{\lim} = 0$ *and the origin* $Z_2 = 0$ *is globally asymptotically stable.*

Proof Computing the time derivative of V, using the last $n-1$ equations in (62) and the skew-symmetry of the system matrix, one obtains:

$$\dot{V} = \frac{\partial V}{\partial Z_2}\dot{Z}_2 = Z_2^T\,[S(u_1)Z_2 + b\,w_2] = \frac{1}{\prod_{j=1}^{n-2} k_j} z_n w_2.$$

Hence, using the control law (64)

$$\dot{V} = -\frac{k_{w_2}(u_1)}{\prod_{j=1}^{n-2} k_j} z_n^2 \leq 0, \tag{65}$$

which shows that the Lyapunov-like function V is nonincreasing. This in turn implies that $\|Z_2\|$ is bounded uniformly with respect to the initial conditions. Existence and uniqueness of the system solutions also follows.

Since V is nonincreasing and bounded below, it converges to a non-negative limit value $V_{\lim}$. Also, $k_{w_2}(u_1)$ is uniformly continuous as a function of time because $k_{w_2}(\cdot)$ is continuous and $|u_1(t)|$, $|\dot{u}_1(t)|$ are bounded. Hence, the right-hand side of eq. (65) is uniformly continuous along any system solution and, by application of Barbalat's lemma [20], $\dot{V}$ tends to zero. Therefore, $k_{w_2}(u_1)z_n$ tends to zero. This in turn implies, using the properties of the function $k_{w_2}(\cdot)$ and the boundedness of $|u_1(t)|$ and $|z_n(t)|$, that $u_1(t)z_n(t)$ tends to zero.

We can now proceed in a recursive fashion, exploiting the structure of eq. (62). Taking the time derivative of $u_1^2 z_n$, and using the convergence of $u_1 z_n$ to zero, one gets

$$\frac{d}{dt}(u_1^2 z_n) = -k_{n-2}u_1^3 z_{n-1} + o(t), \qquad \text{with } \lim_{t\to+\infty} o(t) = 0. \tag{66}$$

The function $u_1^3 z_{n-1}$ is uniformly continuous along any system solution because its time derivative is bounded. Therefore, in view of eq. (66) and since $u_1^2 z_n$ tends to zero, $d(u_1^2 z_n)/dt$ also tends to zero (by application of a slightly generalized version of Barbalat's lemma). Hence, both $u_1^3 z_{n-1}$ and $u_1 z_{n-1}$ tend to zero. Taking the time derivative of $u_1^2 z_j$ and repeating the above procedure, one concludes that $u_1 z_j$ tends to zero for $j = 2, \ldots, n$. Through the system equations, this in turn implies the convergence of $\dot{Z}_2$ to zero.

Summing up the squared values of $u_1 z_j$ for $j = 2, \ldots, n$, it is clear that also $u_1^2 V$ tends to zero, together with $u_1 V$. From the already established convergence of $V(t)$ to $V_{\lim}$, we have also that $u_1 V_{\lim}$ tends to zero, implying $V_{\lim} = 0$ if $u_1(t)$ does not asymptotically tend to zero. ∎

Once a signal $u_1(t)$ satisfying the hypotheses of Prop. 4.1 has been chosen, we must design a suitable function $k_{w_2}(u_1)$ and select the constants k_j ($j = 1, \ldots, n-2$) appearing in the definition (61) of the skew-symmetric coordinates

z_i $(i = 4, \ldots, n)$ and in the control signal (63). As it is often the case, tuning several control parameters may be rather delicate. However, it is easily verified that, with the particular choice

$$k_{w_2}(u_1) = k'_{w_2}|u_1|, \qquad k'_{w_2} > 0, \tag{67}$$

the control u_2 given by eqs. (63) and (64) coincides with the eigenvalue assignment control (60) associated with the linear time-invariant system (58). More precisely, there is a one-to-one correspondence between the parameters of the two control laws. One can thus apply classical linear control methods to determine these parameters in order to optimize the performance near the point $Z_2 = 0$, as will be illustrated in Sect. 4.1 in the application to the car-like robot.

According to Prop. 4.1, any sufficiently regular input $u_1(t)$ can be used for the regulation of Z_2 to zero, as long as it does not asymptotically tend to zero. This leaves the designer with some degrees of freedom in the choice of this input when addressing a path following problem. For instance, uniform exponential convergence of $\|Z_2\|$ to zero is obtained when $|u_1|$ remains larger than some positive number. Other sufficient conditions for exponential convergence of $\|Z_2\|$ to zero, which do not require u_1 to have always the same sign, may also be derived. For example, if $|\dot{u}_1|$ is bounded, then it is sufficient to have $|u_1|$ periodically larger than some positive number.

Finally, we note that the requirement that the signal $u_1(t)$ does not asymptotically tend to zero can be relaxed. In fact, non-convergence of $u_1(t)$ to zero under the assumption that $|\dot{u}_1|$ is bounded implies that $\int_0^t |u_1(\tau)|d\tau$ tends to infinity with t. When using the control (64) with the choice (67), divergence of this integral is the actual necessary condition for the asymptotic convergence of $\|Z_2\|$ to zero. This appears when the control (64) is interpreted as a stabilizing linear control for the time-invariant system (58) obtained by replacing the time variable by the aforementioned integral. However, this integral may still diverge when $u_1(t)$ tends to zero 'slowly enough' (like $t^{-\frac{1}{2}}$, for example). This indicates that $\|Z_2\|$ may converge to zero even when u_1 does, a fact that will be exploited next.

Point stabilization via smooth time-varying feedback Proposition 4.1 suggests a simple way of determining a smooth time-varying feedback law which globally asymptotically stabilizes the origin $Z = 0$ of the whole system. In this case, the role of the control u_1 is to complement the action of the control w_2 (or, through eq. (63), u_2) in order to guarantee asymptotic convergence of z_1 to zero as well.

Proposition 4.2. *Consider the same control of Prop. 4.1*

$$w_2 = -k_{w_2}(u_1)z_n,$$

complemented with the following time-varying control

$$u_1 = -k_{u_1} z_1 + \eta(Z_2, t), \qquad k_{u_1} > 0, \tag{68}$$

where $\eta(Z_2, t)$ *is a uniformly bounded and class* C^{p+1} *function* ($p \geq 1$) *with respect to time, with all successive partial derivatives also uniformly bounded with respect to time, and such that:*

C1: $\eta(0, t) = 0$, $\forall t$;

C2: *There exist a time-diverging sequence* $\{t_i\}$ ($i = 1, 2, \ldots$) *and a positive continuous function* $\alpha(\cdot)$ *such that*

$$\|Z_2\| \geq l > 0 \quad \Longrightarrow \quad \sum_{j=1}^{p} \left(\frac{\partial^j \eta}{\partial t^j}(Z_2, t_i) \right)^2 \geq \alpha(l) > 0, \quad \forall i.$$

Under the above controls, the origin $Z = 0$ *is globally asymptotically stable.*

Proof It has already been shown that the positive function $V(Z_2)$ used in Prop. 4.1 is nonincreasing along the closed-loop system solutions, implying that $\|Z_2\|$ is bounded uniformly with respect to initial conditions.

The first equation of the controlled system is

$$\dot{z}_1 = -k_{u_1} z_1 + \eta(Z_2, t). \tag{69}$$

This is the equation of a stable linear system subject to the bounded additive perturbation $\eta(Z_2, t)$. Therefore, existence and uniqueness of the solutions is ensured, and $|z_1|$ is bounded uniformly with respect to initial conditions.

From the expression of u_1, and using the regularity properties of $\eta(Z_2, t)$, it is found that u_1 is bounded along the solutions of the closed-loop system, together with its first derivative. Therefore, Prop. 4.1 applies; in particular, $V(Z_2)$ tends to some positive limit value $V_{\lim}$, $\|\dot{Z}_2(t)\|$ tends to zero, and $Z_2(t)$ tends to zero if $u_1(t)$ does not.

We proceed now by contradiction. Assume that $u_1(t)$ does not tend to zero. Then, $\|Z_2(t)\|$ tends to zero. By uniform continuity, and in view of condition C1, $\eta(Z_2, t)$ also tends to zero. Equation (69) becomes then a stable linear system subject to an additive perturbation which asymptotically vanishes. As a consequence, $z_1(t)$ tends to zero implying, by the expression of u_1, that so does also $u_1(t)$, yielding a contradiction. Therefore, $u_1(t)$ must asymptotically tend to zero.

Differentiating the expression of u_1 with respect to time, and using the convergence of $u_1(t)$ and $\|\dot{Z}_2(t)\|$ to zero, we get

$$\dot{u}_1(t) = \frac{\partial \eta}{\partial t}(Z_2(t), t) + o(t), \qquad \text{with } \lim_{t \to +\infty} o(t) = 0.$$

Since $(\partial\eta/\partial t)(Z_2,t)$ is uniformly continuous (its time derivative is bounded), both $\dot{u}_1(t)$ and $(\partial\eta/\partial t)(Z_2,t)$ converge to zero (Barbalat's lemma). By using similar arguments, one can also show that the total time derivative of $(\partial\eta/\partial t)(Z_2(t),t)$ and $(\partial^2\eta/\partial t^2)(Z_2(t),t)$ tend to zero. By repeating the same procedure as many times as necessary, one obtains that $(\partial^j\eta/\partial t^j)(Z_2,t)$ $(j = 1,\ldots,p)$ tends to zero. Hence,

$$\lim_{t\to\infty} \sum_{j=1}^{p} \left(\frac{\partial^j \eta}{\partial t^j}(Z_2(t),t) \right)^2 = 0. \tag{70}$$

Assume now that $V_{\lim}$ is different from zero. This would imply that $\|Z_2(t)\|$ remains larger that some positive real number l (which can be calculated from $V_{\lim}$). Eq. (70) is then incompatible with the condition C2 imposed on the function $\eta(Z_2,t)$. Hence, $V_{\lim}$ is equal to zero and Z_2 asymptotically converges to zero. Then, by uniform continuity and using condition C1, $\eta(Z_2,t)$ also tends to zero. Finally, in view of the expression of u_1, asymptotic convergence of z_1 to zero follows immediately. ∎

We point out that controls u_1 and u_2 resulting from Prop. 4.2 are smooth with respect to the state provided that the functions $\eta(Z_2,t)$ and $k_{w_2}(u_1)$ are themselves smooth. On the other hand, if $k_{w_2}(u_1)$ is chosen as in eq. (67), u_2 is only continuous.

In the overall controller, the choices related to u_2 (or w_2) can be made along the same lines indicated at the end of Sect. 4.1. In particular, the same control law (60) based on input scaling can be used. As for u_1, the gain k_{u_1} is typically chosen on the basis of an approximate linearization at the origin. Its second component $\eta(Z_2,t)$, which introduces an explicit time dependence, is referred to as the *heat function* in order to establish a parallel with probabilistic global minimization methods. The role of $\eta(Z_2,t)$ in the control strategy is fundamental, for it 'forces motion' until the system has not reached the desired configuration, thus preventing the state from converging to other equilibrium points.

The conditions imposed by Prop. 4.2 on the heat function η can be easily met. For example, the three following functions

$$\eta_1(Z_2,t) = \|Z_2\|^2 \sin t \tag{71}$$

$$\eta_2(Z_2,t) = \sum_{j=0}^{n-2} a_j \sin(\beta_j t)\, z_{2+j} \tag{72}$$

$$\eta_3(Z_2,t) = \sum_{j=0}^{n-2} a_j \frac{\exp(b_j z_{2+j}) - 1}{\exp(b_j z_{2+j}) + 1} \sin(\beta_j t), \tag{73}$$

satisfy the conditions whenever $a_j \neq 0$, $b_j \neq 0$, $\beta_j \neq 0$, and $\beta_i \neq \beta_j$ for $i \neq j$. For the first function, this is obvious. For the second function, the proof can be found in [43]. The same proof basically applies to the third function, which has the additional feature of being uniformly bounded with respect to all its arguments. It should be noted that it is not strictly necessary to use time-periodic functions.

The choice of a suitable heat function is critical for the overall control performance. In general, it is observed that functions (72) and (73) behave better than (71) with respect to the induced asymptotic convergence rate. For the last two functions, the parameters a_j and b_j (which characterize the 'slope' of $\eta_2(Z_2,t)$ and $\eta_3(Z_2,t)$ near the origin $Z_2 = 0$) have much influence on the transient time needed for the solutions to converge to zero.

Application to the car-like robot For the $(2,4)$ chained form (10) that pertains to the car-like robot, the non-trivial part of the change of coordinates (61) is defined by

$$z_4 = k_1\chi_2 + \chi_4,$$

since we have from eq. (56)

$$L_{h_1} z_3 = L_{h_1}\chi_3 = \chi_4.$$

The skew-symmetric form (62) becomes in this case

$$\begin{aligned}
\dot{z}_1 &= u_1 \\
\dot{z}_2 &= u_1 z_3 \\
\dot{z}_3 &= -k_1 u_1 z_2 + u_1 z_4 \\
\dot{z}_4 &= -k_2 u_1 z_3 + w_2,
\end{aligned}$$

with

$$w_2 = (k_1 + k_2)u_1 z_3 + u_2.$$

In view of Prop. 4.1 and eq. (67), the control input u_2 for the skew-symmetric form is chosen as

$$\begin{aligned}
u_2 &= -k'_{w_2}|u_1|z_4 - (k_1 + k_2)u_1 z_3 \\
&= -k_1 k'_{w_2}|u_1|\chi_2 - (k_1 + k_2)u_1\chi_3 - k'_{w_2}|u_1|\chi_4. \qquad (74)
\end{aligned}$$

The value of the three gains k_1, k_2, and k'_{w_2} can be selected on the basis of the aforementioned correspondence between the structure of eq. (74) and the

eigenvalue assignment control (60). In particular, by comparing the expression of u_2 with the input-scaled version (30) of the linear tracking controller, which assigns three coincident eigenvalues in $-\alpha|u_1|$ (with $\alpha > 0$), we can solve for the three gains as

$$k_1 = \alpha^2/3, \qquad k_2 = 8\alpha^2/3, \qquad k'_{w_2} = 3\alpha.$$

In this association, one should remember that $\chi_2 = x_4$, $\chi_3 = x_3$, and $\chi_4 = x_2$. In particular, the following gain parameters have been used

$$k_1 = 1/3, \quad k_2 = 8/3, \quad k_{w_2} = 3,$$

corresponding to three eigenvalues in -1 for the input-scaled linear approximation.

As for the control input u_1, which is given by eq. (68), we have set $k_{u_1} = 10$, corresponding to an eigenvalue in -10 for the linear approximation of the x-error dynamics, and we have used the heat function η_2 with the following parameters

$$\begin{aligned} a_0 &= 40, \ a_1 = 20, \ a_2 = 20, \\ \beta_0 &= 1, \quad \beta_1 = 2, \quad \beta_2 = 3. \end{aligned}$$

The above controller has been simulated for a car-like robot with $\ell = 1$ m executing a parallel parking maneuver. The desired configuration is the origin of the state space, while the initial configuration at $t_0 = 0$ is

$$x(0) = 0, \quad y(0) = -5, \quad \theta(0) = 0, \quad \phi(0) = 0.$$

Figures 20–26 show respectively the cartesian motion of the vehicle, the time evolution of x, y, θ and ϕ, and the actual commands v_1 and v_2 applied to the car-like robot, obtained from u_1 and u_2 via the chained-form input transformation (9).

After performing several other numerical tests, we can conclude that:

- The motion is quite natural in the first phase of approaching.
- For any stabilization task, the final part of the motion resembles a parallel parking maneuver.
- Basically, the larger are the a_j parameters of the heat function η_2, the shorter is the transient time. On the other hand, more control effort is required far from the goal.
- The final convergence close to the goal is rather slow.

In order to achieve practical convergence to a small ball around the origin in finite time, a simpler, discontinuous heat function can be used. For example,

we have chosen

$$\eta_4(z_2, z_3, z_4, t) = \begin{cases} k_\eta \sin t & \text{if} z_2^2 + z_3^2 + z_4^2 \geq \epsilon \\ 0 & \text{if} z_2^2 + z_3^2 + z_4^2 < \epsilon, \end{cases}$$

with $\epsilon = 10^{-3}$, $k_\eta = 20$, and modified one of the previous gains by setting $k_{u_1} = 5$. The obtained results are illustrated in Figs. 27–33. The norm of the final cartesian error is equal to $3.35 \cdot 10^{-2}$ m (only due to the y-coordinate), while the final values of θ and ϕ are $2.5 \cdot 10^{-3}$ rad and $5.5 \cdot 10^{-3}$ rad, respectively. This condition is reached in about 17 sec.

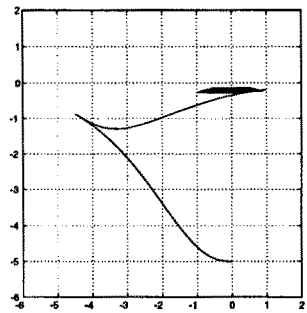


Fig. 20. Point stabilization with time-varying feedback and heat function η_2: cartesian motion

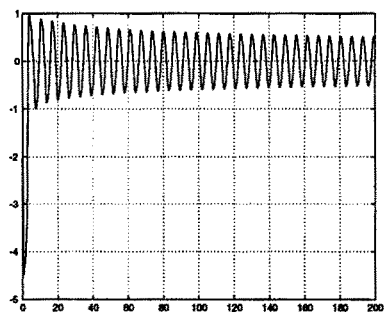


Fig. 21. Point stabilization with time-varying feedback and heat function η_2: x (m) vs. time (sec)

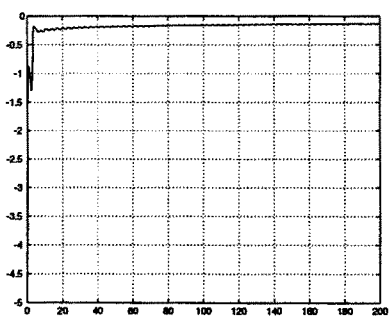


Fig. 22. Point stabilization with time-varying feedback and heat function η_2: y (m) vs. time (sec)

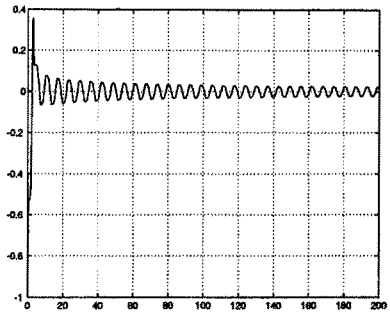


Fig. 23. Point stabilization with time-varying feedback and heat function η_2: θ (rad) vs. time (sec)

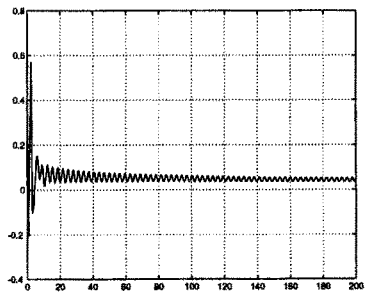


Fig. 24. Point stabilization with time-varying feedback and heat function η_2: ϕ (rad) vs. time (sec)

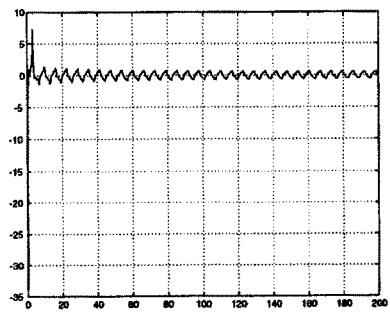


Fig. 25. Point stabilization with time-varying feedback and heat function η_2: v_1 (m/sec) vs. time (sec)

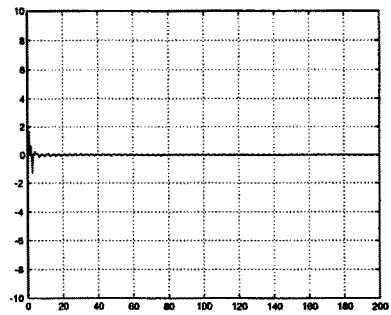


Fig. 26. Point stabilization with time-varying feedback and heat function η_2: v_2 (rad/sec) vs. time (sec)

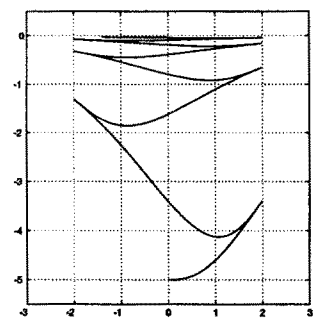


Fig. 27. Point stabilization with time-varying feedback and heat function η_4: cartesian motion

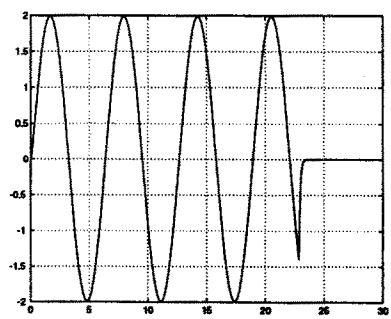


Fig. 28. Point stabilization with time-varying feedback and heat function η_4: x (m) vs. time (sec)

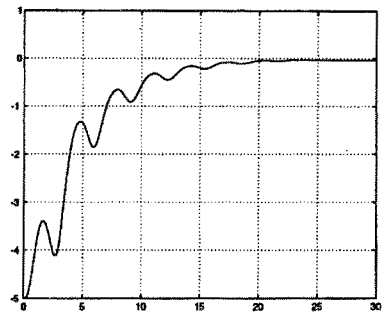


Fig. 29. Point stabilization with time-varying feedback and heat function η_4: y (m) vs. time (sec)

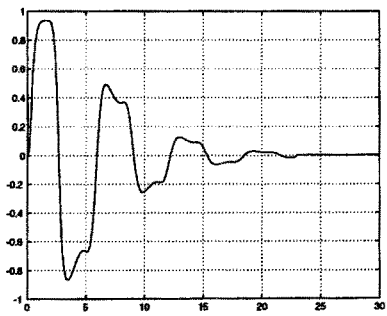


Fig. 30. Point stabilization with time-varying feedback and heat function η_4: θ (rad) vs. time (sec)

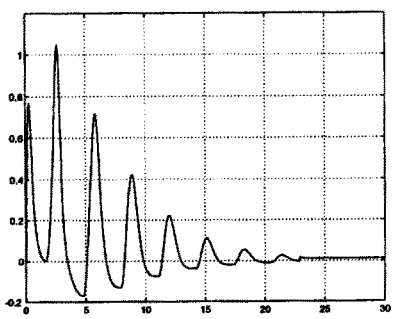


Fig. 31. Point stabilization with time-varying feedback and heat function η_4: ϕ (rad) vs. time (sec)

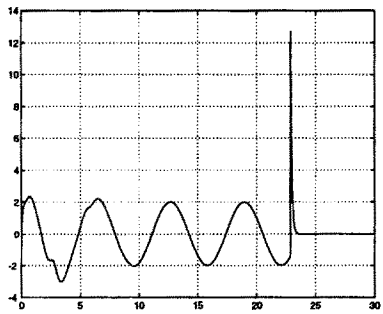


Fig. 32. Point stabilization with time-varying feedback and heat function η_4: v_1 (m/sec) vs. time (sec)

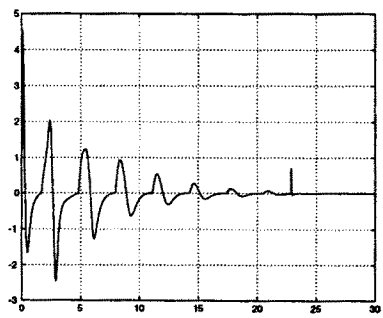


Fig. 33. Point stabilization with time-varying feedback and heat function η_4: v_2 (rad/sec) vs. time (sec)

4.2 Control via nonsmooth time-varying feedback

We present next the design of a nonsmooth time-varying feedback controller that stabilizes the chained form to the origin. The contents of this section are based on [48] and [49]. The idea of decomposing the system into two parts and sequentially defining the two control inputs is very similar to the one pursued in Sect. 4.1. In fact, also this technique provides as a byproduct a solution to the path following problem. The fundamental difference here is that the feedback control law will depend, in addition to the exogenous time variable, on a piecewise-constant function of the state. Moreover, the actual construction of the control law for a subvector of dimension $n-1$ of the state is based on the so-called backstepping method.

Preliminaries We begin with two definitions.

Definition 1 A function $h : \mathbb{R}^+ \mapsto \mathbb{R}^+$ is said to be of *class* $\mathcal{K}$ if it is strictly increasing and such that $h(0) = 0$. ◁

Definition 2 For a nonlinear time-varying system

$$\dot{x} = f(x,t), \qquad x \in Q \subset \mathbb{R}^n, \quad t \geq t_0, \tag{75}$$

the equilibrium point x_e is globally $\mathcal{K}$-*exponentially stable* if there exists a positive constant λ (independent of t_0) and a function $h(\cdot)$ of class $\mathcal{K}$ such that all solutions $x(t)$ of eq. (75) satisfy

$$\|x(t) - x_e\| \leq h(\|x(t_0) - x_e\|)\, e^{-\lambda(t-t_0)}, \quad \forall x(t_0) \in Q,\ \forall t \geq t_0. \qquad \triangleleft$$

We note that a $\mathcal{K}$-exponentially stable system is uniformly asymptotically stable and, in addition, has an exponential rate of convergence. However, exponential stability in the sense of Lyapunov is stronger than the above property, because it involves a special function of class $\mathcal{K}$ which is linear, i.e.,

$$h(\|x(t_0) - x_e\|) = \bar{h}\|x(t_0) - x_e\|,$$

with $\bar{h} > 0$ independent from t_0 and $x(t_0)$. Nevertheless, the two properties are equivalent with respect to the rate of convergence, once the initial condition $x(t_0)$ is given.

The following technical lemma establishes sufficient conditions for obtaining exponential convergence in a time-varying system. Its proof can be found in [48].

Lemma 4.3. *Consider a scalar nonlinear time-varying system*

$$\dot{x} = -a(x,t)x + d(x,t), \qquad t \geq t_0, \tag{76}$$

under the following assumptions:

- *there exists a solution* $x(t)$ *of eq. (76) for any* $x(t_0)$ *and* $t \geq t_0$*;*
- $a(x,t)$ *is such that for all* $x(t)$

$$\left| \int_{t_0}^{t} (a(x(\tau),\tau) - \mu)\, d\tau \right| \leq P, \quad \forall t \geq t_0,$$

where μ *and* P *are some positive constants;*
- $d(x,t)$ *is bounded for all* $x(t)$ *as*

$$|d(x(\tau),\tau)| \leq De^{-\gamma(t-t_0)}, \quad \forall t \geq t_0,$$

for some positive constants D *and* γ*.*

Then,

$$|x(t)| \leq c\,(|x(t_0)| + D)e^{-(\beta-\varepsilon)(t-t_0)}, \quad \forall \varepsilon > 0,$$

where $\beta = \min\{\mu,\gamma\} > 0$ *and* $c = \max\{e^P, e^{2P}/\varepsilon\}$*.*

Backstepping control design For convenience, we rewrite here the chained system (7)

$$\begin{aligned} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_2 u_1 \\ &\vdots \\ \dot{x}_n &= x_{n-1} u_1, \end{aligned} \tag{77}$$

and partition its state as $X = (x_1, X_2)$, with $X_2 = (x_2, x_3, \ldots, x_n)$.

As previously noted, when u_1 is a predefined function of time, X_2 satisfies a linear time-varying equation driven by the input u_2. In the following, we will assume that a structure is assigned for the signal $u_1(t)$ and address the design of a feedback control law for u_2 so as to make $X_2 = 0$ a $\mathcal{K}$-exponentially stable equilibrium point. Later, we will add the variable x_1 to the picture and choose the specific form of control u_1 so as to obtain $\mathcal{K}$-exponential stability of $X = 0$ for the complete system.

The structure of u_1 is chosen by combining the simplicity of an open-loop command, which is updated as a function of the state only at discrete time instants, with the benefits of adding a time-varying exogenous signal. In particular, let a sequence of equidistant time instants $\{t_0, t_1, t_2, \ldots\}$ be defined as

$$t_h = hT, \qquad T = t_{h+1} - t_h > 0,$$

and define the control u_1 as

$$u_1(t) = k(X(t_h))f(t), \qquad \text{for } t \in [t_h, t_{h+1}). \tag{78}$$

This structure implies that the input is a function of the state X at time $t = t_h$, while during the interval (t_h, t_{h+1}) it is defined in an open-loop fashion. For the time being, no restrictions are put on the form of the function $k(\cdot)$ beside its boundedness. On the other hand, some assumptions are needed for function $f(t)$.

A1: $f(t) \in C^\infty$;
A2: $0 \leq f(t) \leq 1, \ \forall t \geq t_0$;
A3: $f(t_h) = 0, \ \forall t_h \in \{t_0, t_1, \dots\}$;
A4: $\left| \int_{t_h}^{t} \left(f^{2(j-2)+1}(\tau) - \mu_j \right) d\tau \right| \leq P_j, \ \forall j \in \{3, \dots, n\}, \ \forall t_h \in \{t_0, t_1, \dots\}$,

where μ_j and P_j are positive constants.

Assumption A3 implies that $u_1(t_h) = 0$ for all $t_h \in \{t_0, t_1, \dots\}$, and thus guarantees continuity of $u_1(t)$ with respect to time, even if function k is non-smooth with respect to the state. Assumption A4 is more technical and is used to guarantee controllability of the linear time-varying subsystem and for proving exponential convergence of X_2 to zero by means of Lemma 4.3.

A simple periodic function satisfying the above assumptions is

$$f(t) = \frac{1 - \cos \omega t}{2}, \quad \omega = \frac{2\pi}{T}. \tag{79}$$

This function has a nonzero mean value, a fact which turns out to be important in order to have some 'control energy' sustaining the robot motion as long as an error is present, i.e., $X \neq 0$ (see the related remarks in Sect 4.1). However, $f(t)$ is not restricted to be periodic.

Using eq. (78), the lower part of system (77) becomes

$$\dot{X}_2 = \begin{bmatrix} \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} u_2 \\ k(X(t_h))f(t)x_2 \\ \vdots \\ k(X(t_h))f(t)x_{n-2} \\ k(X(t_h))f(t)x_{n-1} \end{bmatrix}, \quad t \in [t_h, t_{h+1}), \ h = 0, 1, \dots . \tag{80}$$

In the following, we will often write $k = k(X(t_h))$ for compactness.

A feedback law for u_2 that renders $X_2 = 0$ exponentially convergent to zero (in fact, $\mathcal{K}$-exponentially stable) is now derived based on a *backstepping* method—a general technique for controlling systems in cascaded form [22].

Take the last equation in (80) and regard the variable x_{n-1} as a 'dummy' input to be used for driving exponentially the state x_n to its target $x_n^d = 0$. To this end, x_{n-1} should behave as a desired reference signal x_{n-1}^d which is chosen to satisfy

$$kf(t)x_{n-1}^d = -\lambda_n f^{2(n-2)+1}(t)x_n,$$

with arbitrary $\lambda_n > 0$, or equivalently

$$x_{n-1}^d = -\frac{\lambda_n}{k} f^{2(n-2)}(t)x_n. \tag{81}$$

This choice is convenient because, if $x_{n-1} = x_{n-1}^d$, the last equation in (80) becomes

$$\dot{x}_n = -\lambda_n f^{2(n-2)+1}(t)x_n.$$

By virtue of assumption A4, we may use Lemma 4.3 (with $a(t) = \lambda_n f^{2(n-2)+1}(t)$ and $d(t) = 0$) to infer that, at least with the nominal dummy input, x_n exponentially converges to zero. The convergence rate depends on the choice of the parameter λ_n.

During a transient phase, we will actually have a difference $\tilde{x}_{n-1} = x_{n-1} - x_{n-1}^d \neq 0$ leading to

$$\dot{x}_n = kf(t)x_{n-1}^d + kf(t)\tilde{x}_{n-1} = -\lambda_n f^{2(n-2)+1}(t)x_n + kf(t)\tilde{x}_{n-1}.$$

We can use again Lemma 4.3 (with $a(t) = \lambda_n f^{2(n-2)+1}(t)$ and $d(t) = kf(t)\tilde{x}_{n-1}$) to conclude that x_n exponentially converges to zero, provided that $|kf(t)\tilde{x}_{n-1}|$ is exponentially converging to zero as well. This will be guaranteed by the remaining steps of the recursive procedure.

Consider now the next to last equation in (80) and regard the variable x_{n-2} as the new dummy input, to be used for driving the state x_{n-1} to its target x_{n-1}^d specified by eq. (81). To obtain exponential convergence of $\tilde{x}_{n-1}$ to zero, x_{n-2} should behave as a desired reference x_{n-2}^d which is chosen to satisfy

$$kf(t)x_{n-2}^d = -\lambda_{n-1} f^{2(n-3)+1}(t)(x_{n-1} - x_{n-1}^d) + \dot{x}_{n-1}^d,$$

with arbitrary $\lambda_{n-1} > 0$. In fact, if $x_{n-2} = x_{n-2}^d$, the next to last equation in (80) gives

$$\dot{\tilde{x}}_{n-1} = -\lambda_{n-1} f^{2(n-3)+1}(t)\tilde{x}_{n-1},$$

and we can use again Lemma 4.3 to show that $\tilde{x}_{n-1}$ exponentially converges to zero, with rate depending on the parameter λ_{n-1}. This holds also when $\tilde{x}_{n-2} =$

$x_{n-2} - x_{n-2}^d \neq 0$, provided that $|kf(t)\tilde{x}_{n-2}|$ is itself converging exponentially to zero.

Backstepping further, x_{n-3} will be regarded as the dummy input in the second to last equation in (80), and the same control design is repeated until we reach the top equation, in which the true input command u_2 is present. In this last step, the control input will be defined as

$$u_2 = -\lambda_2(x_2 - x_2^d) + \dot{x}_2^d. \tag{82}$$

With this choice, one can show (under additional assumptions on the function $k(\cdot)$ to be detailed later) that x_2 will converge exponentially to x_2^d, which in turn implies that x_3 converges exponentially to x_3^d, and so on until it is proven that x_n converges exponentially to $x_n^d = 0$.

At the end of this procedure, a reference value has been defined for each state component of X_2, namely

$$\begin{aligned} x_n^d &= 0 \\ x_{j-1}^d k(X(t_h)) f(t) &= -\lambda_j f^{2j-3}(t)(x_j - x_j^d) + \dot{x}_j^d, \quad \lambda_j > 0, \quad j = 3, \dots, n. \end{aligned} \tag{83}$$

By expanding the time derivatives in eq. (83), the above reference values become a combination, weighted by powers of $f(t)$, of the state components in X_2. For example, in the case of the (2,4) chained form pertaining to a car-like robot we would obtain:

$$\begin{aligned} x_4^d &= 0 \\ x_3^d &= -\lambda_4 f^4(t) x_4 / k(X(t_h)) \\ x_2^d &= \left[-\lambda_3 f^3(t)(x_3 - x_3^d)/k(X(t_h)) + \dot{x}_3^d / k(X(t_h))\right] / f(t) \\ &= -\left[\left(\lambda_3 f^2(t) + \lambda_4 f^4(t)\right) / k(X(t_h))\right] x_3 \\ &\quad - \left[\left(\lambda_3 \lambda_4 f^6(t) + 4\lambda_4 f^2(t)\right) / k^2(X(t_h))\right] x_4. \end{aligned}$$

Note that the order of the exponent of function $f(t)$ in eq. (81) has been chosen large enough to guarantee that $f(t) = 0$ implies all $x_i^d = 0$. This choice forces the car-like robot to align its orientation and steering angle with the x-axis whenever it performs a cusp during the motion. In fact, the control u_1 may change sign only in correspondence to instants t in which $f(t) = 0$.

In order to obtain a compact expression for the control input u_2, the reference values (83) can be reorganized and written as

$$\begin{aligned} x_i^d &= f^{2(i-1)}(t) \sum_{j=i+1}^{n} \frac{g_{ij}}{k^{j-i}(X(t_h))} x_j, \qquad i = 2, \dots, n-1, \\ x_n^d &= 0, \end{aligned} \tag{84}$$

where the functions $g_{ij} = g_{ij}(f, \dot{f}, \ldots, f^{(j-i-1)})$ are smooth with respect to their arguments and are defined recursively by

$$\begin{aligned} g_{n-1,n} &= -\lambda_n \\ g_{i-1,j} &= g_{ij}\left[\lambda_i f^{2(i-1)} + 2(i-1)\dot{f}\right] + f(\dot{g}_{ij} + g_{i,j+1} f) \\ g_{i-1,i} &= -\lambda_i + f^2 g_{i,i+1} \\ g_{ip} &= 0, \qquad\qquad \text{if } p \leq i \text{ or } p = n+1, \end{aligned} \tag{85}$$

for $i, j = 2, \ldots, n$, being

$$\dot{g}_{ij} = \sum_{m=0}^{j-i-1} \frac{\partial g_{ij}}{\partial f^{(m)}} f^{(m+1)}.$$

Summarizing eqs. (78), (82), (84), and (85), the following control structure is obtained

$$u_1 = k(X(t_h))f(t) \tag{86}$$

$$u_2 = \Gamma^T(k(X(t_h)), t)\, X_2, \tag{87}$$

with

$$\Gamma^T(k,t) = [\Gamma_2(k,t) \ \ldots \ \Gamma_n(k,t)]$$

and

$$\begin{aligned} \Gamma_2(k,t) &= -\lambda_2 + f^3 g_{23} \\ \Gamma_j(k,t) &= \frac{f\left(\lambda_2 f g_{2j} + 2\dot{f} g_{2j} + f\dot{g}_{2j} + f^2 g_{2,j+1}\right)}{k^{j-2}(X(t_h))}, \quad j = 3, \ldots, n. \end{aligned}$$

Convergence results The main convergence results for the above controller are now presented. We start by providing conditions on $k(\cdot)$ so as to guarantee exponential convergence to zero of the state X_2 in eq. (80), a result which provides a solution for the path following problem.

Proposition 4.4. *Consider system (80), where u_2 is given by eq. (87) and $f(t)$ has the properties A1–A4. Assume further that:*

- $k(0) = 0$;
- $X_2 \neq 0$ *implies* $k(X) \neq 0$;
- *there exists a constant* K *such that* $|k(X)| \leq K$, $\forall X \in \mathbb{R}^n$;
- *whenever* $|k(X(t_h))| < K$, *it is*

$$|k(X(t_h))| \geq \kappa_j |\tilde{x}_j(t)|^{\frac{1}{2(n-2)}}, \quad \forall j = 3, \ldots, n, \tag{88}$$

where $\tilde{x}_j = x_j - x_j^d$ *and* κ_j *is a positive constant.*

Then, $X_2 = 0$ is $\mathcal{K}$-exponentially stable, i.e., there exist a constant $\lambda_{X_2} > 0$ and a function $h_{X_2}(\cdot, T)$ of class $\mathcal{K}$ such that

$$\|X_2(t)\| \leq h_{X_2}(\|X_2(t)\|, T)e^{-\lambda_{X_2}(t-t_0)}, \quad \forall X_2(t_0) \in \mathbb{R}^{n-1}, \forall t \geq t_0.$$

Proof (sketch of) The first three assumptions are used to prove that the 'error' variables $\tilde{x}_2, \ldots, \tilde{x}_n$ converge to zero, i.e., each x_i $(i = 2, \ldots, n)$ converges to its reference value x_i^d defined by eq. (83). Lemma 4.3 is the main tool in this analysis, resulting in an exponential rate of convergence which can also be estimated. Then, one can show that the original states x_j $(j = 2, \ldots, n)$ can be expressed as a weighted sum of the error variables $\tilde{x}_r$ $(r = j+1, \ldots, n)$, essentially by reversing the construction of the x_i^d in eq. (84). Finally, by using eq. (88), $\mathcal{K}$-exponential stability of $X_2 = 0$ is obtained. ∎

A function $k(\cdot)$ satisfying the assumptions of the above proposition is given by

$$k(X) = \mathrm{sat}\left(-\beta\left[x_1 + \mathrm{sgn}(x_1)G\left(\|X_2\|\right)\right], K\right), \tag{89}$$

where

$$\mathrm{sat}(z, K) = \begin{cases} z & \text{if} |z| \leq K, \\ K\mathrm{sgn}(z) & \text{if} |z| > K, \end{cases}$$

and

$$\mathrm{sgn}(z) = \begin{cases} 1 & \text{if } z \geq 0, \\ -1, & \text{if } z < 0, \end{cases}$$

$$G\left(\|X_2\|\right) = \kappa \|X_2\|^{\frac{1}{2(n-2)}},$$

$$\beta = 1 / \int_{t_h}^{t_{h+1}} f(\tau) d\tau,$$

with κ a positive constant.

By using Prop. 4.4, one can finally establish global $\mathcal{K}$-exponential stability of the origin $X = 0$ of the total system in chained form (77), thus solving the point stabilization problem. We give this result without the proof, which is rather long and can be found in [49].

Proposition 4.5. *Consider system (77), where u_1 is given by eq. (86), with $k(X)$ chosen as in eq. (89) and $f(t)$ satisfying assumptions A1–A4, and u_2 is given by eq. (87). Then, $X = 0$ is $\mathcal{K}$-exponentially stable, i.e., there exist a constant $\lambda_X > 0$ and a function $h_X(\cdot, T)$ of class $\mathcal{K}$ such that*

$$\|X(t)\| \leq h_X(\|X(t_0)\|, T)e^{-\lambda_X(t-t_0)}, \quad \forall X(t_0) \in \mathbb{R}^n, \forall t \geq t_0.$$

Note the following facts.

- It can be shown that the class $\mathcal{K}$-function $h_X(\cdot, T)$ is not Lipschitz around the origin. In particular, its derivative tends to infinity when $\|X(t_0)\|$ approaches zero.
- The exponential convergence rate λ_X in Prop. 4.5 can be made arbitrarily fast by choosing β in eq. (89) and $\lambda_2, \ldots, \lambda_n$ in eq. (85) large enough. However, the time needed to drive x_1 to an arbitrarily small neighborhood of zero cannot be less than T. As a consequence, the class $\mathcal{K}$-function $h_X(\cdot, T)$ increases exponentially with the 'period' T. On the other hand, reducing T may result in a large control effort for some initial conditions.
- In the generic time interval $[t_h, t_{h+1})$, the control input u_1 is essentially open-loop being only a function of the state at time t_h, whereas the control input u_2 is a true feedback, for it depends continuously on the state variables X_2.

Application to the car-like robot For the car-like robot in (2,4) chained form, we present here explicit formulas for generating u_2 according to eq. (87). Let λ_2, λ_3, and λ_4 be three positive constants. Choose $f(t)$ as in eq. (79) and $k(X)$ as in eq. (89). We have

$$t_{h+1} - t_h = T = \frac{2\pi}{\omega} \qquad \text{and} \qquad \beta = \frac{1}{\int_{t_h}^{t_{h+1}} f(\tau) d\tau} = \frac{\omega}{\pi}, \quad \forall h.$$

Equations (85) give

$$\begin{aligned} g_{23} &= -\lambda_3 - \lambda_4 f^2 \\ \dot{g}_{23} &= -2\lambda_4 f \dot{f} \\ g_{24} &= -\lambda_4(\lambda_3 f^4 + 4\dot{f}) \\ \dot{g}_{24} &= -4\lambda_3\lambda_4 f^3 \dot{f} - 4\lambda_4 \ddot{f} \\ g_{25} &= 0, \end{aligned}$$

to be used in

$$\begin{aligned} \Gamma_2 &= -\lambda_2 + f^3 g_{23} \\ \Gamma_3 &= f\left[\lambda_2 f g_{23} + 2\dot{f} g_{23} + f\dot{g}_{23} + f^2 g_{24}\right] / k(X(t_h)) \\ \Gamma_4 &= f\left[\lambda_2 f g_{24} + 2\dot{f} g_{24} + f\dot{g}_{24} + f^2 g_{25}\right] / k^2(X(t_h)). \end{aligned}$$

The control input u_1 is provided by eq. (86).

The following parameters have been used in the various functions that define the control laws (86) and (87):

$$K = 2, \quad \omega = 1, \quad \kappa = 3, \quad \lambda_2 = \lambda_3 = \lambda_4 = 1.$$

Note that the first control input may switch only every 2π sec, i.e., at $t \in \{0, 2\pi, 4\pi, \dots\}$.

The above controller has been simulated for a car-like robot with $\ell = 1$ m executing a parking maneuver. The desired configuration is the origin of the state space, while the initial configuration at $t_0 = 0$ is

$$x(0) = -1, \quad y(0) = -1, \quad \theta(0) = -\pi/4, \quad \phi(0) = 0. \tag{I}$$

Figures 34–40 show respectively the cartesian motion of the vehicle, the time evolution of x, y, θ and ϕ, and the actual commands v_1 and v_2 applied to the car-like robot, obtained from u_1 and u_2 through the chained-form input transformation (9). Similarly to the smooth time-varying controller of Sect. 4.1, the generated cartesian motion is natural and resembles a parallel parking maneuver in the final phase. Convergence to the desired configuration appears to be faster; however, $x = x_1$ converges much slower than the other variables y, θ and ϕ, which are related to $X_2 = (x_2, x_3, x_4)$. This behavior can be predicted by using Lemma 4.3.

These observations have been confirmed also by other simulations. For example, Figs. 41–47 show the results obtained by using the same controller in order to execute a reorientation maneuver. The desired configuration is again the origin of the state space, while the initial configuration is

$$x(0) = 0, \quad y(0) = 0, \quad \theta(0) = \pi/6, \quad \phi(0) = 0. \tag{II}$$

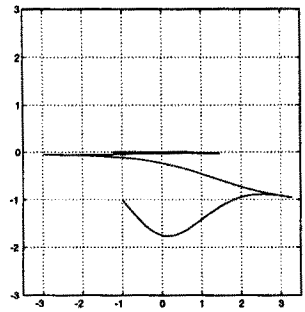


Fig. 34. Point stabilization with nonsmooth time-varying feedback (I): cartesian motion

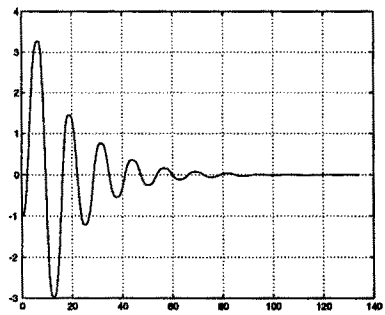


Fig. 35. Point stabilization with nonsmooth time-varying feedback (I): x (m) vs. time (sec)

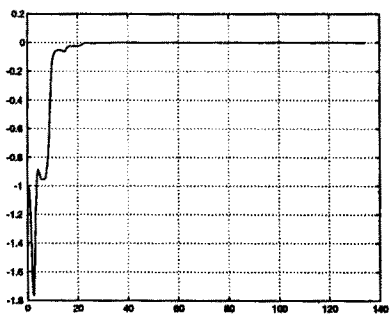


Fig. 36. Point stabilization with nonsmooth time-varying feedback (I): y (m) vs. time (sec)

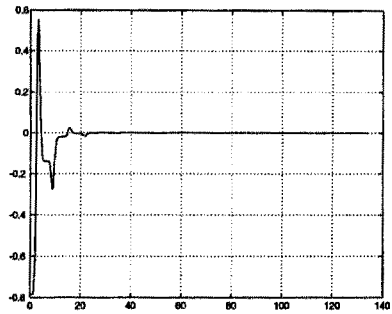


Fig. 37. Point stabilization with nonsmooth time-varying feedback (I): θ (rad) vs. time (sec)

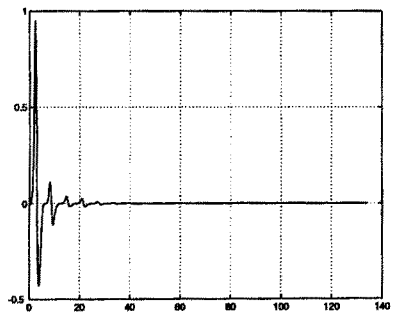


Fig. 38. Point stabilization with nonsmooth time-varying feedback (I): ϕ (rad) vs. time (sec)

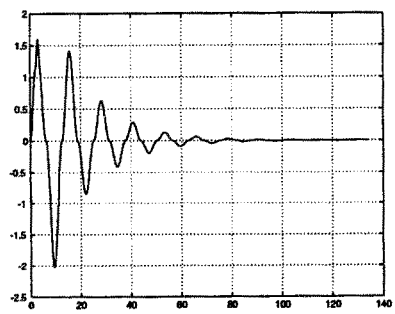


Fig. 39. Point stabilization with nonsmooth time-varying feedback (I): v_1 (m/sec) vs. time (sec)

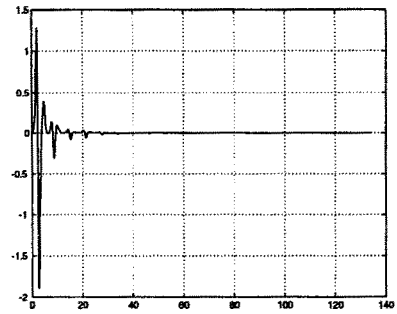


Fig. 40. Point stabilization with nonsmooth time-varying feedback (I): v_2 (rad/sec) vs. time (sec)

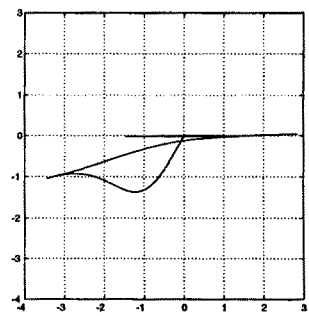


Fig. 41. Point stabilization with nonsmooth time-varying feedback (II): cartesian motion

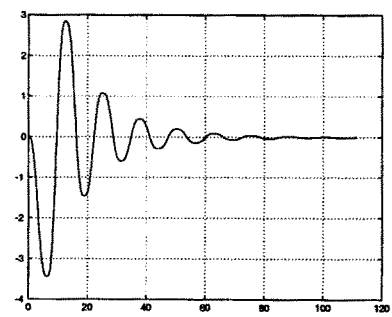


Fig. 42. Point stabilization with nonsmooth time-varying feedback (II): x (m) vs. time (sec)

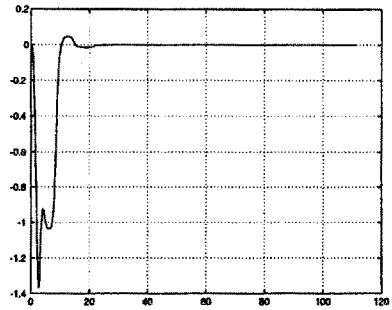


Fig. 43. Point stabilization with nonsmooth time-varying feedback (II): y (m) vs. time (sec)

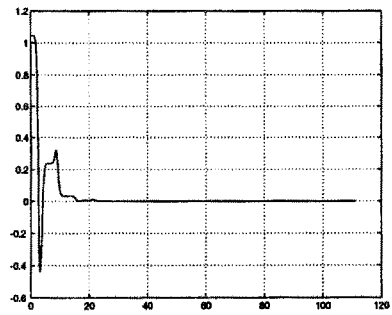


Fig. 44. Point stabilization with nonsmooth time-varying feedback (II): θ (rad) vs. time (sec)

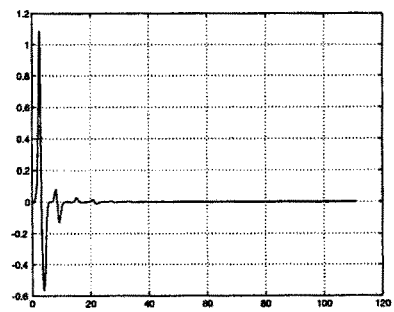


Fig. 45. Point stabilization with nonsmooth time-varying feedback (II): ϕ (rad) vs. time (sec)

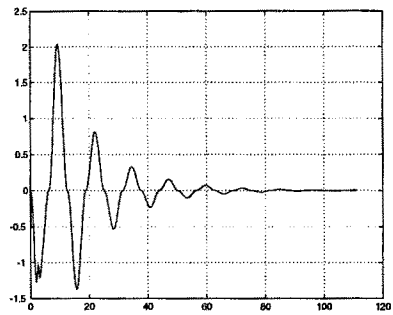


Fig. 46. Point stabilization with nonsmooth time-varying feedback (II): v_1 (m/sec) vs. time (sec)

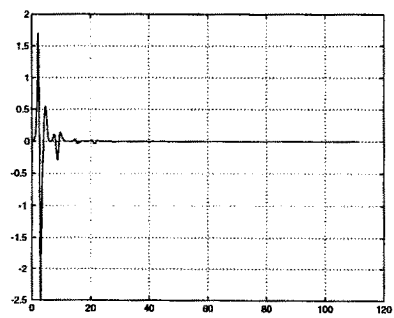


Fig. 47. Point stabilization with nonsmooth time-varying feedback (II): v_2 (rad/sec) vs. time (sec)

4.3 About exponential convergence

The peculiar convergence behavior of both presented stabilizing methods deserves some comments. We have already pointed out in Sect. 2.2 that the failure of the linear controllability test for the car-like robot indicates that smooth exponential stability in the sense of Lyapunov cannot be obtained. Recall that (local) exponential stability means that the system trajectories $X(t)$ satisfy the following inequality

$$\|X(t)\| \leq K\|X(t_0)\|e^{-\lambda(t-t_0)}, \qquad \forall X(t_0) \in B, \quad \forall t \geq t_0, \tag{90}$$

with K, λ positive real numbers and B a neighborhood of the origin. The practical significance of this relationship is twofold: *(i)* small initial errors cannot produce arbitrarily large transient deviations since $\|X(t)\| \leq K\|X(t_0)\|$, and *(ii)* all solutions converge to zero exponentially.

While it is still unclear whether both properties can be simultaneously achieved for nonholonomic systems, one can still design a control law that guarantees at least one of the two. In the case of smooth time-varying feedback laws, such as the one presented in Sect. 4.1, it may be easily verified that

$$\|X(t)\| \leq K\|X(t_0)\|, \qquad \forall X(t_0), \quad \forall t \geq t_0, \tag{91}$$

holds for some positive constant K. However, when using the control law w_2 of Prop. 4.2, convergence to zero of $\|Z\|$ (and hence, of $\|X\|$) cannot be exponential. In fact, if this were the case, u_1 would itself converge to zero exponentially, and thus the integral $\int_0^t |u_1(\tau)|d\tau$ would not diverge. This is in contradiction with the fact that divergence of this integral is necessary for the asymptotic convergence of $\|Z_2\|$ to zero. As a matter of fact, it is only possible to show that

$$\|X(t)\| \leq K\|X(t_0)\|\rho(t), \qquad \text{with} \quad \rho(0) = 1, \ \lim_{t\to\infty} \rho(t) = 0, \tag{92}$$

where $\rho(t)$ is a decreasing function whose convergence rate is strictly less than exponential. This theoretical expectation is confirmed by the simulations results of Sect. 4.1. In particular, it has been observed [41] that smooth time-varying feedback control applied to a unicycle yields a convergence rate slower than $t^{-1/2}$ for most initial configurations, a fact that can be proven using center manifold theory.

On the other hand, existing nonsmooth feedback laws for nonholonomic systems do not guarantee uniform boundedness of the transient error ratio $\|X(t)\|/\|X(t_0)\|$. For example, the piecewise-continuous time-invariant feedback law proposed in [8] for the stabilization of a unicycle yields

$$\|X(t)\| \leq (K_1 + K_2\|X(t_0)\|)e^{-\lambda(t-t_0)}, \qquad \forall X(t_0), \quad \forall t \geq t_0, \tag{93}$$

with K_1, K_2 positive real numbers. All solutions converge to zero exponentially, but a small initial error or perturbation may produce transient deviations whose size is larger than some constant.

Similarly, we have seen that the nonsmooth time-varying feedback of Sect. 4.2 guarantees $\mathcal{K}$-exponential stability for general chained-form systems. Even if all solutions converge to zero exponentially, this type of asymptotic stability is weaker than property (90), in the sense that small initial errors or perturbations can produce transient deviations of much larger amplitude. Nevertheless, it is stronger than (93), for such deviations are not bounded below by some positive constant.

The above discussion may suggest that smooth time-varying feedback laws are somewhat less sensitive to initial errors than nonsmooth feedback laws. This degree of robustness is paid in terms of the asymptotic rate of convergence, which is not exponential. However, smooth time-varying feedback may be modified to achieve *practical* exponential stability, in the sense that the system state may be steered to any desired small neighborhood of the origin in arbitrary time. This fact is illustrated by the simulation results obtained with the heat function η_4 in Sect. 4.1.

5 Conclusions

We have presented and compared several feedback solutions for point stabilization, path following and trajectory tracking control tasks executed by a mobile robot with car-like kinematics.

The problem of accurate tracking of a persistent trajectory can be solved using either linear control synthesis, based on the approximate linearization of the system around the nominal trajectory, or nonlinear (static or dynamic) control synthesis, achieving exact linearization of the (input-output or full-state) closed-loop equations. Local exponential convergence to zero tracking error is obtained in the linear case, while global exponential convergence with prescribed error dynamics is guaranteed in the nonlinear case. In both approaches, the closed-loop controller consists of a nominal feedforward term and of an error feedback action.

For the stabilization to a fixed configuration, the use of new classes of time-varying nonlinear controllers has proven to be effective. From a theoretical point of view, time-varying feedback overcomes the obstruction on the existence of smooth time-invariant stabilizing control laws for nonholonomic systems. Two types of time-varying control laws were presented, respectively expressed by a smooth and a nonsmooth function of the robot state. In both cases, we have recognized that path following can be formulated as a subproblem of point stabilization. The asymptotic rate of convergence of the smooth controller is

lower than the exponential one obtained in the nonsmooth case. However, it may be questioned whether the theoretical convergence rate alone is a good measure of the overall control performance. In practice, what really matters is a rapid initial decay of the error to a small neighborhood of zero under realistic experimental conditions.

The reported numerical simulations have shown the benefit of feedback control in recovering from initial errors with respect to the desired fixed or moving target. In order to fully appreciate these results, we remark that errors (at the initial time or later) can be interpreted as the effect of an instantaneous disturbance acting on the system. Therefore, the robot motion under feedback control is robust with respect to such non-persistent disturbances.

Most of the results have been presented using a canonical transformation of the system into chained form. Although the use of chained forms is not needed in principle, it allows to obtain systematic results that can be extended beyond the considered case study of a car-like mobile robot. For example, the control results hold true also for a car towing N trailers, each attached at the midpoint of the rear axle of the previous one (zero hooking). On the other hand, the control problem for the general case of N trailers with nonzero hooking is still open, because a chained-form transformation is not available for this system.

Throughout this study, we have dealt with a first-order kinematic model of the mobile robot, in which velocities were assumed to be the control inputs. Extension to second-order kinematics, with accelerations as inputs, and inclusion of vehicle dynamics, with generalized forces as inputs, are possible. In particular, we point out that the nominal dynamics of the vehicle can be completely canceled by means of a nonlinear state feedback so as to obtain a second-order, purely kinematic problem.

Concerning the application of the proposed feedback controllers to real mobile robot systems, there are several non-ideal conditions that may affect the actual behavior of the controlled robot, notably: uncertain kinematic parameters of the vehicle (including, e.g., the wheels' radius); mechanical limitations such as backlash at the steering wheels and limited range of the steering angle; actuator saturation and dead-zone; noise and biases in the transformation from physical sensor data to the robot state; quantization errors in a digital implementation. Control robustness with respect to these kinds of uncertainties and/or disturbances is an open and challenging subject of research. For linear as well as nonlinear systems, Lyapunov exponential stability implies some degree of robustness with respect to perturbations. However, since this kind of stability has not been demonstrated for the point stabilization problem of nonholonomic systems, the connection between robustness properties and asymptotic (even exponential) rate of convergence is not yet well understood.

It should also be noted that perturbations acting on nonholonomic mobile robots are not of equal importance, depending on which component of the state

is primarily affected. A deviation in a direction compatible with the vehicle mobility (e.g., sliding of the wheels on the ground) is clearly not as severe as a deviation which violates the kinematic constraints of the system (e.g., lateral skidding of the car-like robot). In any case, proprioceptive sensors may not reveal these perturbing actions and all the controllers presented in this chapter—which assume that the exact robot state is available—would fail in completing their task. A possible solution would be to close the feedback loop using exteroceptive sensor measurements, which provide absolute information about the robot location in its workspace. Currently, it is not clear whether the best solution would be to estimate the robot state from these measurements and then use the previous controllers, or to design new control laws aimed at zeroing the task error directly at the sensor-space level.

6 Further reading

In addition to the references cited to support the results so far presented, many other related works have appeared in the literature. Hereafter, we mention some of the most significant ones.

A detailed reference on the kinematics of wheeled mobile robots is [2]. The dynamics of general nonholonomic systems was thoroughly analyzed in [31]. A controllability study for kinematic models of car-like robots with trailers was presented in [24], while stabilizability results for both kinematic and dynamic models of nonholonomic systems were given in [5,7].

The problem of designing input commands that drive a nonholonomic mobile robot to a desired configuration has been first addressed through open-loop techniques. Purely differential-geometric approaches were followed in [23,50], while the most effective solutions have been obtained by resorting to chained-form transformations and sinusoidal steering [28], or by using piecewise-constant functions as control inputs [26]. In [36] it was shown how the existence of differentially flat outputs can be exploited in order to design efficiently open-loop controls.

A number of works have dealt with the problem of controlling via feedback the motion of a unicycle. In fact, both discontinuous and time-varying feedback controllers were first proposed and analyzed for this specific kinematics. The trajectory tracking problem was solved in [39] by means of a local feedback action. Use of dynamic feedback linearization was proposed in [14]. A piecewise-continuous feedback with an exponential rate of convergence was presented in [8] for the point stabilization task, and later extended to the path following problem in [47]. Another piecewise-continuous controller, obtained through an appropriate switching sequence, was devised in [5]. The explicit inclusion of the exogenous time variable in a smooth feedback law was proposed in [38].

In [34], a hybrid stabilization strategy was introduced that makes use of a time-invariant feedback law far from the destination and of a time-varying law in its vicinity. The use of a discontinuous transformation in polar coordinates allowing to overcome the limitation of Brockett's theorem was independently proposed in [1] and [3] for the point stabilization problem; strictly speaking, these schemes are not proven to be stable in the sense of Lyapunov, for they only ensure exponential convergence of the error to zero. A survey of control techniques for the unicycle can be found in [9].

For car-like robots, the trajectory tracking problem was also addressed in [13] through the use of dynamic feedback linearization, and in [16] via flat outputs design and time-scaling. Path following via input scaling was proposed in [15,37]. As for the point stabilization problem, the successful application of time-varying feedback to the case of car-like robots [43] has subsequently motivated basic research work aimed at exploring the potentialities of this approach. In particular, results have been obtained for the whole class of controllable driftless nonlinear systems in [11,12], while general synthesis procedures were given in [33] for chained-form systems and in [35] for power-form systems; in the latter, the use of a nonsmooth but continuous time-varying feedback guarantees exponential convergence to the desired equilibrium point. Using an analysis based on homogeneous norms, similar results were obtained for driftless systems in [30], and for chained-form systems in [27] by means of a backstepping technique. Other related works include [17] and [51]. In the first, the problem of approximating a holonomic path via a nonholonomic one is solved by using time-periodic feedback control. In the second, the open-loop sinusoidal steering method is converted to a stabilization strategy, by adding to the nominal command a mixed discontinuous/time-varying feedback action.

Very few papers have explicitly addressed robustness issues in the control of nonholonomic systems. The robustness of a particular class of nonsmooth controllers based on invariant manifolds was analyzed in [10]. Robust stabilization of car-like robots in chained form was obtained in [4] and [25] by applying iteratively a contracting open-loop controller; exponential convergence to the desired equilibrium is obtained for small model perturbations. Another possible approach to the design of effective control laws in the presence of nonidealities and uncertainties is represented by learning control, as shown in [32].

Finally, the design of sensor-level controllers for nonholonomic mobile robots is at the beginning stage. The general concept of task-driven feedback control for holonomic manipulators is described in [40]. A first attempt to extend this idea to the point stabilization problem of a mobile robotic system can be found in [52].

References

1. M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino, "Closed loop steering of unicycle-like vehicles via Lyapunov techniques," *IEEE Robotics & Automation Mag.*, vol. 2, no. 1, pp. 27–35, 1995.
2. J. C. Alexander and J. H. Maddocks, "On the kinematics of wheeled mobile robots," *Int. J. of Robotics Research*, vol. 8, no. 5, pp. 15–27, 1989.
3. A. Astolfi, "Exponential stabilization of a mobile robot," *3rd European Control Conf.*, Roma, I, pp. 3092–3097, 1995.
4. M. K. Bennani and P. Rouchon, "Robust stabilization of flat and chained systems," *3rd European Control Conf.*, Roma, I, pp. 2642–2646, 1995.
5. A. M. Bloch, M. Reyhanoglu, and N. H. McClamroch, "Control and stabilization of nonholonomic dynamic systems," *IEEE Trans. on Automatic Control*, vol. 37, no. 11, pp. 1746–1757, 1992.
6. R. W. Brockett, "Asymptotic stability and feedback stabilization," in *Differential Geometric Control Theory*, R. W. Brockett, R. S. Millman, H. J. Sussmann (Eds.), Birkhäuser, Boston, MA, pp. 181–191, 1983.
7. G. Campion, B. d'Andrea-Novel, and G. Bastin, "Modeling and state feedback control of nonholonomic mechanical systems," *30th IEEE Conf. on Decision and Control*, Brighton, UK, pp. 1184–1189, 1991.
8. C. Canudas de Wit and O. J. Sørdalen, "Exponential stabilization of mobile robots with nonholonomic constraints," *IEEE Trans. on Automatic Control*, vol. 37, no. 11, pp. 1791–1797, 1992.
9. C. Canudas de Wit, H. Khennouf, C. Samson, and O. J. Sørdalen, "Nonlinear control design for mobile robots," in *Recent Trends in Mobile Robots*, Y. F. Zheng (Ed.), World Scientific Publisher, 1993.
10. C. Canudas de Wit and H. Khennouf, "Quasi-continuous stabilizing controllers for nonholonomic systems: Design and robustness considerations," *3rd European Control Conf.*, Roma, I, pp. 2630–2635, 1995.
11. J.-M. Coron, "Global asymptotic stabilization for controllable systems without drift," *Mathematics of Control, Signals, and Systems*, vol. 5, pp. 295–312, 1992.
12. J.-M. Coron, "Links between local controllability and local continuous stabilization," *2nd IFAC Symp. on Nonlinear Control System Design*, Bordeaux, F, pp. 477–482, 1992.
13. B. d'Andrea-Novel, G. Bastin, and G. Campion, "Dynamic feedback linearization of nonholonomic wheeled mobile robots," *1992 IEEE Int. Conf. on Robotics and Automation*, Nice, F, pp. 2527–2532, 1992.
14. A. De Luca and M. D. Di Benedetto, "Control of nonholonomic systems via dynamic compensation," *Kybernetica*, vol. 29, no. 6, pp. 593–608, 1993.
15. E. D. Dickmanns and A. Zapp, "Autonomous high speed road vehicle guidance by computer vision," *10th IFAC World Congr.*, München, D, pp. 221–226, 1987.
16. M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Design of trajectory stabilizing feedback for driftless flat systems," *3rd European Control Conf.*, Roma, I, pp. 1882–1887, 1995.
17. L. Gurvits and Z. Li, "Smooth time-periodic feedback solutions for nonholonomic motion planning," in *Nonholonomic Motion Planning*, Z. Li, J. Canny (Eds.), Kluwer Academic Publishers, Norwell, MA, pp. 53–108, 1992.

18. A. Isidori, *Nonlinear Control Systems*, 3rd Edition, Springer-Verlag, London, UK, 1995.
19. T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
20. H. K. Khalil, *Nonlinear Systems*, Macmillan, New York, NY, 1992.
21. I. V. Kolmanovsky, M. Reyhanoglu, and N. H. McClamroch, "Discontinuous feedback stabilization of nonholonomic systems in extended power form," *33rd IEEE Conf. on Decision and Control*, Lake Buena Vista, FL, pp. 3469–3474, 1994.
22. M. Krstić, I. Kanellakopoulos, and P. Kokotović, *Nonlinear and Adaptive Control Design*, John Wiley & Sons, New York, NY, 1995.
23. G. Lafferriere and H. J. Sussmann, "Motion planning for controllable systems without drift," *1991 IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, pp. 1148–1153, 1991.
24. J.-P. Laumond, "Controllability of a multibody mobile robot," *IEEE Trans. on Robotics and Automation*, vol. 9, no. 6, pp. 755–763, 1993.
25. P. Lucibello and G. Oriolo, "Stabilization via iterative state steering with application to chained-form systems," *35th IEEE Conf. on Decision and Control*, Kobe, J, pp. 2614–2619, 1996.
26. S. Monaco and D. Normand-Cyrot, "An introduction to motion planning under multirate digital control," *31st IEEE Conf. on Decision and Control*, Tucson, AZ, pp. 1780–1785, 1992.
27. P. Morin and C. Samson, "Time-varying exponential stabilization of chained systems based on a backstepping technique," *35th IEEE Conf. on Decision and Control*, Kobe, J, pp. 1449–1454, 1996.
28. R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Trans. on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.
29. R. M. Murray, "Control of nonholonomic systems using chained forms," *Fields Institute Communications*, vol. 1, pp. 219–245, 1993.
30. R. M. Murray and R. T. M'Closkey, "Converting smooth, time-varying, asymptotic stabilizers for driftless systems to homogeneous, exponential stabilizers," *3rd European Control Conf.*, Roma, I, pp. 2620–2625, 1995.
31. J. I. Neimark and F. A. Fufaev, *Dynamics of Nonholonomic Systems*, American Mathematical Society, Providence, RI, 1972.
32. G. Oriolo, S. Panzieri, and G. Ulivi, "An iterative learning controller for nonholonomic robots," *1996 IEEE Int. Conf. on Robotics and Automation*, Minneapolis, MN, pp. 2676–2681, 1996.
33. J.-B. Pomet, "Explicit design of time-varying stabilizing control laws for a class of controllable systems without drift," *Systems & Control Lett.*, vol. 18, pp. 147–158, 1992.
34. J.-B. Pomet, B. Thuilot, G. Bastin, and G. Campion, "A hybrid strategy for the feedback stabilization of nonholonomic mobile robots," *1992 IEEE Int. Conf. on Robotics and Automation*, Nice, F, pp. 129–134, 1992.
35. J.-B. Pomet and C. Samson, "Time-varying exponential stabilization of nonholonomic systems in power form," INRIA Rep. 2126, Dec. 1993.
36. P. Rouchon, M. Fliess, J. Lévine, and P. Martin, "Flatness and motion planning: The car with n trailers," *2nd European Control Conf.*, Gröningen, NL, pp. 1518–1522, 1993.

37. M. Sampei, T. Tamura, T. Itoh, and M. Nakamichi, "Path tracking control of trailer-like mobile robot," *1991 IEEE/RSJ Int. Work. on Intelligent Robots and Systems*, Osaka, J, pp. 193–198, 1991.
38. C. Samson, "Velocity and torque feedback control of a nonholonomic cart," in *Advanced Robot Control*, C. Canudas de Wit (Ed.), Birkhäuser, Boston, MA, pp. 125–151, 1991.
39. C. Samson and K. Ait-Abderrahim, "Feedback control of a nonholonomic wheeled cart in cartesian space," *1991 IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, pp. 1136-1141, 1991.
40. C. Samson, M. Le Borgne, B. Espiau, *Robot Control: The Task Function Approach*," Oxford Science Publications, Oxford, UK, 1991.
41. C. Samson and K. Ait-Abderrahim, "Feedback stabilization of a nonholonomic wheeled mobile robot, *1991 IEEE/RSJ Int. Work. on Intelligent Robots and Systems*," Osaka, J, pp. 1242-1247, 1991.
42. C. Samson, "Path following and time-varying feedback stabilization of a wheeled mobile robot," *2nd Int. Conf. on Automation, Robotics and Computer Vision*, Singapore, 1992.
43. C. Samson, "Time-varying feedback stabilization of car-like wheeled mobile robots," *Int. J. of Robotics Research*, vol. 12, no. 1, pp. 55–64, 1993.
44. C. Samson, "Control of chained systems. Application to path following and time-varying point-stabilization of mobile robots," *IEEE Trans. on Automatic Control*, vol. 40, no. 1, pp. 64–77, 1995.
45. E. D. Sontag, "Feedback stabilization of nonlinear systems," in *Robust Control of Linear Systems and Nonlinear Control*, M. A. Kaashoek, J. H. van Schuppen, A. C. M. Ran (Eds.), Birkhäuser, Cambridge, MA, pp. 61–81, 1990.
46. O. J. Sørdalen, "Conversion of the kinematics of a car with n trailers into a chained form," *1993 IEEE Int. Conf. on Robotics and Automation*, Atlanta, GA, vol. 1, pp. 382–387, 1993.
47. O. J. Sørdalen and C. Canudas de Wit, "Exponential control law for a mobile robot: Extension to path following," *IEEE Trans. on Robotics and Automation*, vol. 9, no. 6, pp. 837–842, 1993.
48. O. J. Sørdalen, "Feedback control of nonholonomic mobile robots," Ph. D. Thesis, The Norwegian Institute of Technology, Trondheim, NO, Mar. 1993.
49. O. J. Sørdalen and O. Egeland, "Exponential stabilization of nonholonomic chained systems," *IEEE Trans. on Automatic Control*, vol. 40, no. 1, pp. 35–49, 1995.
50. H. J. Sussmann and W. Liu, "Limits of highly oscillatory controls and the approximation of general paths by admissible trajectories," Tech. Rep. SYCON-91-02, Rutgers University, Feb. 1991.
51. A. R. Teel, R. M. Murray, and G. Walsh, "Nonholonomic control systems: From steering to stabilization with sinusoids," *31st IEEE Conf. on Decision and Control*, Tucson, AZ, pp. 1603–1609, 1992.
52. D. Tsakiris, C. Samson, and P. Rives, "Vision-based time-varying stabilization of a mobile manipulator," *6th Int. Conf. on Control, Automation, Robotics and Vision*, Singapore, 1996.

Probabilistic Path Planning

P. Švestka and M. H. Overmars

Utrecht University

1 Introduction

The robot path planning problem, which asks for the computation of collision free paths in environments containing obstacles, has received a great deal of attention in the last decades [25,15]. In the basic problem, there is one robot present in a static and known environment, and the task is to compute a collision-free path describing a motion that brings the robot from its current position to some desired goal position. Variations and extensions of this basic problem statement are numerous.

To start with, for a large class of robots (i.e., nonholonomic robots) computation of collision-free paths is not sufficient. Not only are the admissible robot placements constrained by obstacles and the robot geometry, but also are the directions of motion subject to constraints. For example, mobile robots moving on wheels have such nonholonomic constraints, due to the fact that their wheels are not allowed to slide. Another realistic scenario is that of multiple robots acting in the same environment. In this case, apart from the restrictions imposed by the obstacles, robot geometry, and possible nonholonomic constraints, one also has to avoid collisions between the robots mutually. Moving obstacles, uncertainties in sensing, and inexact control add further levels of difficulty.

In order to build robots that can autonomously act in real-life environments, path planning problems as sketched above need to be solved. However, it has been proven that, in general, solving even the basic path planning problem requires time exponential in the robots number of degrees of freedom. In spite of this discouraging problem complexity, various such *complete* planners have been proposed. Their high complexity however makes them impractical for most applications. And every extension of the basic path planning problem adds in computational complexity. For example, if we have n robots of d degrees of freedom each, the complexity becomes exponential in nd. Or if we allow for moving obstacles, the problem becomes exponential in their number [9,35]. Assuming uncertainties in the robots sensing and control, leads to an exponential dependency on the complexity of the obstacles [9].

The above bounds deal with the exact problem, and therefore apply to *complete planners*. These are planners that solve any solvable problem, and return failure for each non-solvable one. So for most practical problems it seems

impossible to use such complete planners. This has lead many researchers to consider simplifications of the problem statement.

A quite recent direction of research, which we just want to mention briefly here, deals with the formulation of assumptions on the robot environment that reduce the path planning complexity. This is based on the belief that there exists a substantial gap between the theoretical worst-case bounds of path planning algorithms and their practical complexity. A number of researchers have attempted to formulate assumptions on the obstacles that prohibit the (artificial) constructions that cause the worst-case bounds. Examples of such assumptions are, amongst others, *fatness* [54,53], *bounded local complexity* [36], and *dispersion* [33]. However, this line of research has been mainly of theoretical nature, and has not yet resulted in implementations of practical path planners. Also, it is currently not clear whether similar results can be obtained for extensions of the basic path planning problem.

Instead of assuming things about the robot environment, many researchers have simply dropped the requirement of completeness for the planner. Heuristic planners have been developed that solve particular difficult problems in impressively low running times. However, the same planners also fail or consume prohibitive time on seemingly simpler ones. For autonomous robots in realistic environments this might be a problem, since one cannot predict the path planning problems such robots will face.

So, on one hand, completeness is a preferred property of motion planners for autonomous robots, while, on the other hand, only heuristic algorithms are capable of solving many of the practical problems that people are interested in. This has lead to the design of path planners that satisfy weaker forms of completeness, in particular *resolution completeness* and *probabilistic completeness*. In this chapter we deal with the latter. A planner is called probabilistically complete if, given a solvable problem, the probability of solving it converges to 1 as the running time goes to infinity. Such a planner is guaranteed to solve any solvable problem within finite time. And if one can estimate the probability of solving a problem with respect to the running time, one has an even stronger result.

Two of the most successful such planners are the *probabilistic path planner* (*PPP*) and the *randomized path planner* (*RPP*).

RPP [5,17] is a potential field planner, that escapes local minima by performing Brownian motions. It has successfully been applied to articulated robots with many degrees of freedom (dof). Also, it has been used for checking whether parts can be removed from aircraft engines for inspection and maintenance, and for automatically synthesising a video clip with graphically simulated human and robot characters entailing a 78-dimensional configuration space. The planner works as follows : Given a goal configuration g, a potential field U is computed, being a function that assigns positive real values to con-

figurations. Roughly, as is the case for potential fields in general, U is defined by an attracting potential of the goal configuration, and repulsing potentials of the obstacles, and can be seen as a landscape with the obstacles as mountains and the goal configuration as lowest point. Connecting the start configuration s to the goal configuration g is attempted by "descending" along U. Such a *down motion* always ends in a local minimum. If this local minimum is g (the global minimum), then the problem is solved. If this is not the case, a Brownian motion is performed, and the process is repeated. Using well-known properties of Brownian motions, *RPP* can be proven to be probabilistically complete [17,24]. Moreover, a calculation of the finite expected number of Brownian motions is given in [24]. This calculation uses the fact that the basins B_i of attraction of the local minima form a partition of the free configuration space. For each pair (B_i, B_j) one can define the transition probability p_{ij} that a Brownian motion, starting at the minimum of B_i, terminates somewhere in B_j. The expected number of Brownian motions can then be expressed as a function of the transition probabilities p_{ij}'s. This nice theoretical result has however the practical drawback that the p_{ij}'s are, in non-trivial cases, unknown. Although *RPP* proves to be very powerful for many practical problems, the method also has some drawbacks. For example, since the planner is potential field based, it does not memories any knowledge about the configuration space after having solved a particular problem, and, for this reason, each new problem requires a whole new search. In other words, it is a single shot approach. Furthermore, it appears to be easy to create seemingly simple problems for which the planner consumes a more than reasonable amount of time, due to very low transition probabilities between certain basins. Also, the method does not apply directly to nonholonomic robots.

Other probabilistically complete planners for static and dynamic environments utilising genetic algorithms are described in [1,7]. Other work on related probabilistic path planning approaches includes [16]. We will not go into details here.

This chapter gives a survey on the probabilistic path planner *PPP*, which is a very general planner, or planning scheme, building probabilistic roadmaps by randomly selecting configurations from the free configuration space and interconnecting certain pairs by simple feasible paths. The method is probabilistically complete and not restricted to any particular class of robots.

A first single-shot version of the planner for free-flying planar robots was described in [31] and subsequently expanded into a general learning approach, for various robot types, in [32]. Independently, "*PPP*-like" preprocessing schemes for holonomic robots where introduced in [21] and [14]. These schemes also build probabilistic roadmaps in the free C-space, but focus on the case of many-dof robots. In [23] the ideas developed in [32] and [21] have been combined, resulting in an even more powerful planner for high-dof robots. Simultaneously, *PPP*

has been applied to nonholonomic robots. Planners for car-like robots that can move both forwards and backwards as well as such that can only move forwards are described in [45,47]. *PPP* applied to tractor-trailer robots is the topic of [49,39]. Probabilistic completeness of the planners for nonholonomic robots is proven in [47]. Recently some first results on the expected running times of *PPP*, under certain geometric assumptions on the free configuration space, have been obtained [22,20,3]. For a thorough survey of probabilistic path planning for holonomic robots we also refer to the thesis of Kavraki [19]. Finally, extensions of *PPP* addressing multi-robot path planning problems have been presented in [46,48].

In this chapter an overview is given of the algorithmic aspects of *PPP* and applications of the planning scheme to various robot types are discussed. Also, some theory is presented regarding probabilistic completeness and expected running times. The chapter is organised as follows: In Section 2 the probabilistic paradigm is described in its general form. In the following two sections the paradigm is applied to specific robot types, i.e., to holonomic robots (free-flying and articulated) in Section 3, and to nonholonomic mobile robots (car-like and tractor-trailer) in Section 4. In both sections the robot specific components of the algorithm are defined, and obtained simulation results are presented. Sections 5 and 6 are of a more theoretical nature. In Section 5 aspects regarding probabilistic completeness of the method are discussed, and proofs of probabilistic completeness are given for the planners described in this chapter. Section 6 deals with analyses of expected running time. Results by Kavraki et al. [22,20,3] are reviewed, and some new results are presented as well. In Section 7 an extension of *PPP* for solving multi-robot path planning problems is described, and simulation results are given for problems involving multiple car-like robots.

2 The Probabilistic Path Planner

The *Probabilistic Path Planner* (*PPP*) can be described in general terms, without focusing on any specific robot type. The idea is that during the *roadmap construction phase* a data structure is incrementally constructed in a probabilistic way, and that this data structure is later, in the *query phase*, used for solving individual path planning problems.

The data-structure constructed during the roadmap construction phase is an undirected graph $G = (V, E)$, where the nodes V are probabilistically generated free configurations and the edges E correspond to (simple) feasible paths. These simple paths, which we refer to as *local paths*, are computed by a *local planner*. A local planner L is simply a function that takes two configurations as arguments, and returns a path connecting them, that is feasible in absence

of obstacles (that is, the path respects the constraints of the robot). Proper choice of the local planner guarantees probabilistic completeness of the global planner, as we will see in Section 5. If, given two configurations a and b, the path $L(a,b)$ is collision-free, then we will say that L *connects* from a to b.

In the query phase, given a start configuration s and a goal configuration g, we try to connect s and g to suitable nodes $\tilde{s}$ and $\tilde{g}$ in V. Then we perform a graph search to find a sequence of edges in E connecting $\tilde{s}$ to $\tilde{g}$, and we transform this sequence into a feasible path. So the paths generated in the query phase (that is described in detail later) are basically just concatenations of local paths, and therefore the properties of these "global paths" are induced by the local planner.

2.1 The roadmap construction phase

We assume that we are dealing with a robot $\mathcal{A}$, and that L is a local planner that constructs paths for $\mathcal{A}$. We assume that L is *symmetric*, that is, for any pair of configurations (a,b) $L(a,b)$ equals $L(b,a)$ reversed. See Section 2.3 for remarks on non-symmetric local planners. As mentioned above, in the roadmap construction phase a probabilistic roadmap is constructed, and stored in an undirected graph $G = (V,E)$. The construction of the roadmap is performed incrementally in a probabilistic way. Repeatedly a random free configuration c is generated and added to V. Heuristics however are used for generating more nodes in "difficult" areas of the free configuration space (or *free C-space*). We try to connect each generated node c to previously added ones with L, and each such successful connection results in a corresponding edge being added to E.

More precisely, this edge adding is done as follows : First, a set N_c of neighbours is chosen from V. This set consists of nodes lying within a certain distance from c, with respect to some distance measure D. Then, in order of increasing distance from c, we pick nodes from N_c. For each such picked node n, we test whether L connects from c to n, and, if so, (c,n) is added to E. However, if n is already graph-connected with c at the moment that it is picked, n is simply ignored. So no cycles can be created and the resulting graph is a forest, i.e., a collection of trees. The motivation for preventing cycles is that no query would ever succeed *thanks to* an edge that is part of a cycle. Hence, adding an edge that creates a cycle can impossibly improve the planners performance in the query phase.

A price to be paid for disallowing cycles in the graph is that in the query phase often unnecessarily long paths will be obtained. Suppose that a and b are two configurations that can easily be connected by some short feasible path. Due to the probabilistic nature of the roadmap construction algorithm, it is very well possible that, at some point, a and b get connected by some very

long path. Obtaining a shorter connection between a and b would require the introduction of a cycle in the graph, which we prevent. So, for any pair of nodes, the first graph path connecting them blocks other possibilities.

There are a number of ways for dealing with this problem. One possibility is to apply an edge adding method that does allow cycles in the graph [32]. These methods however have the disadvantage that they slow down the roadmap construction algorithm, due to the fact that the adding of a node requires more calls of the local planner. Another possibility is to build a forest as described above, but, before using the graph for queries, "smoothing" the graph by adding certain edges that create cycles. Some experiments that we have done indicated that smoothing the graph for just a few seconds significantly reduces the path lengths in the query phase. Finally, it is possible to apply some smoothing techniques on the paths constructed in the query phase. We briefly describe a simple but efficient and general probabilistic path smoothing technique in Section 2.4.

Let $\mathcal{C}$ denote the C-space of the robot, and $\mathcal{C}_f$ the free portion of $\mathcal{C}$ (i.e., the free C-space). To describe the roadmap construction algorithm formally, we need a function $D \in \mathcal{C} \times \mathcal{C} \to \mathbf{R}^+$. It defines the distance measure used, and should give a suitable notion of distance for arbitrary pairs of configurations, taking the properties of the robot $\mathcal{A}$ into account. We assume that D is symmetric. The graph $G = (V, E) \in \mathcal{C}_f \times \mathcal{C}_f^2$ is constructed as follows:

The roadmap construction algorithm

(1) $V = \emptyset$, $E = \emptyset$
(2) **loop**
(3) $\quad c =$ a "randomly" chosen free configuration
(4) $\quad V = V \cup \{c\}$
(5) $\quad N_c =$ a set of neighbours of c chosen from V
(6) $\quad$ **forall** $n \in N_c$, in order of increasing $D(c, n)$ **do**
(7) $\quad\quad$ **if** $\neg$*connected*$(c, n) \wedge L(c, n) \subset \mathcal{C}_f$ **then** $E = E \cup \{(c, n)\}$

The construction algorithm, as described above, leaves a number of choices to be made: A local planner must be chosen, a distance measure must be defined, and it must be defined what the neighbours of a node are. Furthermore, heuristics for generating more nodes in interesting C-space areas should be defined. Some choices must be left open as long as we do not focus on a particular robot type, but certain global remarks can be made here.

Local planner One of the crucial ingredients in the roadmap construction phase is the local planner. As mentioned before, the local planner must construct paths that are *feasible* for $\mathcal{A}$, in absence of obstacles. This simply

means that the paths it constructs describe motions that are performable by the robot, that is, motions that respect the robots constraints. For example, assume the robot is a car-like vehicle moving on wheels, then a local planner that just connects the two argument configurations with a straight-line segment (in C-space) is not suitable, since it describes motions that force the wheels of the robot to slide.

Furthermore, we want the roadmap construction algorithm to be fast. For this, it is important that (1) the local planner constructs its paths in a time efficient manner and (2) the probability that local paths intersect with obstacles is low. The first requirement can be met by keeping the path constructs as simple as possible. For obtaining low intersection probabilities, the local planner should construct paths with relatively small *sweep volumes*. That is, the volumes (in workspace) swept by the robot when moving along local paths should preferably be small. Clearly, local planners minimising these sweep volumes also minimise the probabilities of the local paths intersecting obstacles.

Finally, the local planner should guarantee probabilistic completeness of *PPP*. In Section 5 we give sufficient properties.

Neighbours and edge adding methods Another important choice to be made is that of the neighbours N_c of a (new) node c. As is the case for the choice of the local planner, the definition of N_c has large impact on the performance of the roadmap construction algorithm. Reasons for this are that the choice of the neighbours strongly influences the overall structure of the graph, and that, regardless of how the local planner is exactly defined, the calls of the local planner are by far the most time-consuming operations of the roadmap construction algorithm (due to the collision tests that must be performed).

So it is clear that calls of the local planner that do not effectively extend the knowledge stored in the roadmap should be avoided as much as possible. Firstly, as mentioned before, attempts to connect to nodes that are already in c's connected component are useless. For this reason the roadmap construction algorithm builds a forest. Secondly, local planner calls that fail add no knowledge to the roadmap. To avoid too many local planner failures we only submit pairs of configurations whose relative distance (with respect to D) is small, that is, less than some constant threshold *maxdist*. Thus:

$$N_c \subset \{\tilde{c} \in V | D(c, \tilde{c}) \leq \mathit{maxdist}\} \tag{1}$$

This criterion still leaves many possibilities regarding the actual choice for N_c. We have decided on taking all nodes within distance *maxdist* as neighbours. Experiments with various definitions for N_c on a wide range of problems lead to this choice.

Hence, according to the algorithm outline given above, we try to connect to all "nearby" nodes of c, in order of increasing distance D, but we skip those nodes that are already in c's connected component at the moment that the connection is to be attempted. By considering elements of N_c in this order we expect to maximise the chances of quickly connecting c to other configurations and, consequently, reduce the number of calls to the local planner (since every successful connection results in merging two connected components into one). We refer to the described edge adding method as the *forest method.*

Distance We have seen that a distance function D is used for choosing and sorting the neighbours N_c of a new node c. It should be defined in such a way that $D(a, b)$ (for arbitrary a and b) somehow reflects the chance that the local planner will *fail* to connect a to b. For example, given two configurations a and b, a possibility is to define $D(a, b)$ as the size of the sweep volume (in the workspace) of $L(a, b)$, that is, as the volume of the area swept by the robot when moving along $L(a, b)$. In this way each local planner L induces its own distance measure, that reflects the described "failure-chance" very well. In fact, if the obstacles were randomly distributed points, then this definition would reflect the local planner's failure chance exactly. However, in the general case, exact computations of the described sweep-volumes tend to be rather expensive, and in practice it turns out that certain rough but cheap to evaluate approximations of the sweep volumes are to be preferred.

Node adding heuristics If the number of nodes generated during the roadmap construction phase is large enough, the set V gives a fairly uniform covering of the free C-space. In easy cases, for example for holonomic robots with few degrees of freedom (say not more than 4), G is then well connected. But in more complicated cases where the free C-space is actually connected, G tends to remain disconnected for a long time in certain narrow (and hence difficult) areas of the free C-space.

Due to the probabilistic completeness of the method, we are sure that eventually G will grasp the connectivity of the free space, but to prevent exorbitant running times, it is wise to guide the node generation by heuristics that create higher node densities in the difficult areas. To identify these, there are a number of possibilities.

In some cases, one can use the geometry of the workspace obstacles. For example, for car-like robots adding (extra) configurations that correspond to placements of the robot "parallel" to obstacle edges and "around" convex obstacle corners boosts the performance of the roadmap construction algorithm significantly.

A more general criterion is to use the (run-time) structure of the roadmap G. Given a node $c \in V$, one can count the number of nodes of V lying within some predefined distance of c. If this number is low, the obstacle

region probably occupies a large subset of c's neighbourhood. This suggests that c lies in a difficult area. Another possibility is to look at the distance from c to the nearest connected component not containing c. If this distance is small, then c lies in a region where two components failed to connect, which indicates that this region might be a difficult one (it may also be actually obstructed).
Alternatively, rather than using the structure of the obstacles or the roadmap to identify difficult regions, one can look at the run-time behaviour of the local planner. For example, if the local planner often fails to connect c to other nodes, this is also an indication that c lies in a difficult region. Which particular heuristic function should be used depends to some extent on the input scene.

2.2 The query phase

During the query phase, paths are to be found between arbitrary start and goal configurations, using the graph G computed in the roadmap construction phase. The idea is that, given a start configuration s and a goal configuration g, we try to find feasible paths P_s and P_g, such that P_s connects s to a graph node $\tilde{s}$, and P_g connects g to a graph node $\tilde{g}$, with $\tilde{s}$ graph-connected to $\tilde{g}$ (that is, they lie in the same connected component of G). If this succeeds, we perform a graph search to obtain a path P_G in G connecting $\tilde{s}$ to $\tilde{g}$. A feasible path (in C-space) from s to g is then constructed by concatenating P_s, the subpaths constructed by the local planner when applied to pairs of consecutive nodes in P_G, and P_g reversed. Otherwise, the query fails. The queries should preferably terminate 'instantaneously', so no expensive algorithm is allowed for computing P_s and P_g.

For finding the nodes $\tilde{s}$ and $\tilde{g}$ we use the function $\mathit{query\_mapping} \in \mathcal{C} \times \mathcal{C} \rightarrow V \times V$, defined as follows:

$$\mathit{query\_mapping}(a,b) = (\tilde{a},\tilde{b}), \text{ such that } \tilde{a} \text{ and } \tilde{b} \text{ are connected, and } D(a,\tilde{a}) + D(b,\tilde{b}) = \mathrm{MIN}_{(x,y)\in W} : D(a,x) + D(y,b)$$

$$\text{where } W = \{(x,y) \in V \times V | \mathit{connected}(x,y)\}$$

So $\mathit{query\_mapping}(a,b)$ returns the pair of connected graph nodes $(\tilde{a},\tilde{b})$ that minimise the total distance from a to $\tilde{a}$ and from b to $\tilde{b}$. We will refer to $\tilde{a}$ as a's graph retraction, and to $\tilde{b}$ as b's graph retraction.

The most straightforward way for performing a query with start configuration s and goal configuration g is to compute $(\tilde{s},\tilde{g}) = \mathit{query\_mapping}(s,g)$, and to try to connect with the local planner from s to $\tilde{s}$ and from $\tilde{g}$ to g. However, since no obstacle avoidance is incorporated in the local planner, it

may, in unlucky cases, fail find the connections even if the graph captures the connectivity of free C-space well.

Experiments with different robot types indicated that simple probabilistic methods that repeatedly perform short random walks from s and g, and try to connect to the graph retractions of the end-points of those walks with the local planner, achieve significantly better results. These random walks should aim at maneuvering the robot out of narrow C-space areas (that is, areas where the robot is tightly surrounded by obstacles), and hereby improving the chances for the local planner to succeed. For holonomic robots very good performance is obtained by what we refer to as the *random bounce walk* (see also [32]). The idea is that repeatedly a random direction (in C-space) is chosen, and the robot is moved in this direction until a collision occurs (or time runs out). When a collision occurs, a new random direction is chosen. This method performs much better than for example pure Brownian motion in C-space. For nonholonomic robots walks of a similar nature can be performed, but care must of course be taken to respect the nonholonomic constraints.

2.3 Using a directed graph

In the algorithm outline of *PPP*, as described in the previous section, the computed roadmaps are stored in undirected graphs. For many path planning problems this is sufficient, and it appears that the method is easier and more efficient to implement when based on undirected graphs. For example, path planning problems involving free-flying robots, articulated robots, and (normal) car-like robots can all be dealt with using undirected underlying graphs. There are however path planning problems for which undirected underlying graphs not sufficient, and directed ones are required instead. For example, problems involving car-like robots that can only move forwards require directed underlying graphs.

The existence of an edge (a, b) in the underlying graph G corresponds to the statement that the local planner constructs a feasible path from a to b. If however G is undirected, then the edge contains no information about the direction in which the local planner can compute the path, and, hence, it must correspond to the statement that the local planner constructs a feasible path from a to b, as well as one from b to a. So an edge (a, b) can be added only if the local planner connects in both directions. Doing so, useful information might be thrown away. This will happen in those cases where the local planner connects in exactly one direction, and the fact that it has successfully constructed a feasible path will not be stored. If however the local planner is *symmetric*, which means that it connects from say a to b whenever it connects from b to a, then obviously this problem will never occur. So if the local planner is

symmetric, the underlying graph can be undirected, and if it is not symmetric, then it is better to use a directed graph.

Whether it is possible to implement (good) local planners that are symmetric, depends on the properties of the robot $\mathcal{A}$, defined by the constraints imposed on it.

Definition 4. *A robot $\mathcal{A}$ is* $\mathcal{C}$-symmetric *(*configuration space symmetric*) if and only if any feasible path for $\mathcal{A}$ remains feasible when reversed.*

All holonomic robots are $\mathcal{C}$-symmetric. For nonholonomic robots this is not the case. For example, a car-like robot that can drive forwards as well as backwards is $\mathcal{C}$-symmetric while one that can only drive forwards is not. In terms of control theory, a (nonholonomic) robot is $\mathcal{C}$-symmetric if its control system is symmetric. That is, it can attain a velocity v (in C-space) if and only if it can also attain the velocity $-v$.

Clearly, if $\mathcal{A}$ is $\mathcal{C}$-symmetric, then any local planner L that constructs feasible paths for $\mathcal{A}$ can be made symmetric in a trivial way, by reversing computed paths when necessary. So this implies that for any $\mathcal{C}$-symmetric robot an undirected graph can be used for storing the local paths, and otherwise a directed graph is required.

For directed graphs it is less straightforward to omit the adding of redundant edges than was the case for undirected graphs. We refer to [45] and [47] for discussions on this topic, and sensitive strategies for the adding of directed edges.

2.4 Smoothing the paths

Paths computed in the query phase can be quite ugly and unnecessarily long. This is due to the probabilistic nature of the algorithm, and to the fact that cycle-creating edges are never added.

To improve this, one can apply some path smoothing techniques on these 'ugly' paths. The smoothing routine that we use is very simple. It repeatedly picks a pair of random configurations (c_1, c_2) on the "to be smoothed" path P_C, tries to connect these with a feasible path Q_{new} using the local planner. If this succeeds and Q_{new} is shorter than the path segment Q_{old} in P_C from c_1 to c_2, then it replaces Q_{old} by Q_{new} (in P_C). So basically, randomly picked segments of the path are replaced, when possible, by shorter ones, constructed by the local planner. The longer this is done, the shorter (and nicer) the path gets. Typically, this method smoothes a path very well in less than a second for low dof robots, and in a few seconds for high dof robots.

Still one can argue that this is too much for a query. In that case one must either accept the ugly paths, or use a more expensive edge adding method that builds graphs containing loops. This will result in a slowdown of the roadmap

construction phase, but the gain is that the paths (directly) retrieved in the query phase will be shorter.

3 Application to holonomic robots

In this section an application of *PPP* to two types of holonomic robots is described: free-flying robots and articulated robots.

We consider here only planar holonomic robots. A free-flying robot is represented as a polygon that can rotate and translate freely in the plane among a set of polygonal obstacles. Its C-space is represented by $R^2 \times [0, 2\pi[$. A planar articulated robot $\mathcal{A}$ consists of n *links* $L_1, \ldots, L_n$, which are some solid planar bodies (we use polygons), connected to each other by $n-1$ *joints* $J_2, \ldots, J_n$. Furthermore, the first link L_1 is connected to some *base point* in the workspace by a joint J_1. Each joint is either a *prismatic joint*, or a *revolute joint*. If J_i is a prismatic joint, then link L_i can translate along some vector that is fixed to link L_{i-1} (or to the workspace, if $i = 1$), and if J_i is a revolute joint, then link L_i can rotate around some point that is fixed to link L_{i-1} (or to the workspace, if $i = 1$). The range of the possible translations or rotations of each link L_i is constrained by J_i's *joint bounds*, consisting of a lower bound low_i and an upper bound up_i. The C-space of a n-linked planar articulated robot can, hence, be represented by $[low_1, up_1] \times [low_2, up_2] \times \cdots \times [low_n, up_n]$. In the scenes we show, the revolute joints are indicated by small black discs, and the prismatic joints by small black discs with double arrows.

Since holonomic robots are $\mathcal{C}$-symmetric, it is feasible to use undirected graphs for storing the roadmaps. Some of the (robot specific) details, left open in the discussion of the general method, must be specified.

3.1 Filling in the details

The local planner: A very general local planner exists, that is directly applicable to all holonomic robots. Given two configurations, it connects them by a straight line segment in C-space and checks this line segment for collision and joint limits (if any). We refer to this planner as *the general holonomic local planner*. Collision checking can be done as follows: First, discretise the line segment into a number of configurations $c_1, \ldots, c_m$, such that for each pair of consecutive configurations (c_i, c_{i+1}) no point on the robot, when positioned at configuration c_i, lies further than some ϵ away from its position when the robot is at configuration c_{i+1} (ϵ is a positive constant). Then, for each configuration c_i, test whether the robot, when positioned at c_i and "grown" by ϵ, is collision-free. If none of the m configurations yield collision, conclude that the path is collision-free.

The distance measure: The distance between two configurations a and b is defined as the length (in C-space) of the local path connecting a and b, but scaled in the various C-space dimensions appropriately, in order to reflect the local planners failure chance reasonably. For example, in the case of of a long and thin free flying robot, small variations in orientation (that is, variations in the third dimension) correspond to motions sweeping relatively large volumes in the workspace, and should hence be reflected by large distances, while, on the other hand, for disc-like robots they should be reflected by small distances.

The random walks in the query phase: Section 2.2 described a general scheme for solving a query using a graph constructed in the roadmap construction phase. Multiple random walks were performed from the query configurations s and g, aimed at connecting the end-points of these walks to their graph retractions with the local planner. Remains to define the specific random walks. For holonomic robots, a random bounce walk consists of repeatedly picking at random a direction of motion in C-space and moving in this direction until an obstacle is hit. When a collision occurs, a new random direction is chosen. And so on.

The (maximal) number of these walks (per query) and their (maximal) lengths are parameters of the planner, which we denote by, respectively, N_W and L_W.

Node adding heuristics: For both the free-flying robots as the articulated robots, we utilise the (run-time) structure of G to identify "difficult" areas in which more "random" nodes are to be added than in others. We increase the chances for node generation in areas (of C-space) where the graph shows disconnectivities (that is, where there are a number of separate connected components present).

For high dof robots it also proves helpful to identify nodes lying in difficult areas by considering the success/failure ratio of the local planner. If this ration is low for a particular node (that is, the local planner fails to connect to the node relatively often), this is an indication that the node lies in some difficult area. In this case, more nodes are added in the (near) neighbourhood of the node, in order to locally improve the graph connectivity. We say that the node is *expanded* [21],[23].

3.2 Simulation results

We have implemented the method for planar free-flying and articulated robots in the way described above, and we present some simulation results obtained with the resulting planners. The implementations are in C++ and the experiments were performed on a Silicon Graphics Indigo^2 workstation with an R4400

processor running at 150 MHZ. This machine is rated with 96.5 SPECfp92 and 90.4 SPECint92.

In the test scenes used, the coordinates of all workspace obstacles lie in the unit square. Furthermore, in all scenes we have added an obstacle boundary around the unit square, hence no part of the robot can move outside this square.

The experiments are aimed at measuring the "knowledge" acquired by the method after having constructed roadmaps for certain periods of time. This is done by testing how well the method solves certain (interesting) queries. For each scene S we define a *query test set* $T_Q = \{(s_1, s_1), (s_2, g_2), \ldots, (s_m, g_m)\}$, consisting of a number of configuration pairs (that is, queries). Then, we repeatedly construct a graph for some specified time t, and we count how many of these graphs solve the different queries in T_Q. This experiment is repeated for a number of different construction times t. The results are presented in the tables under the figures. The numbers in the boxes indicate the percentage of the runs that solve the corresponding query within the given time bound.

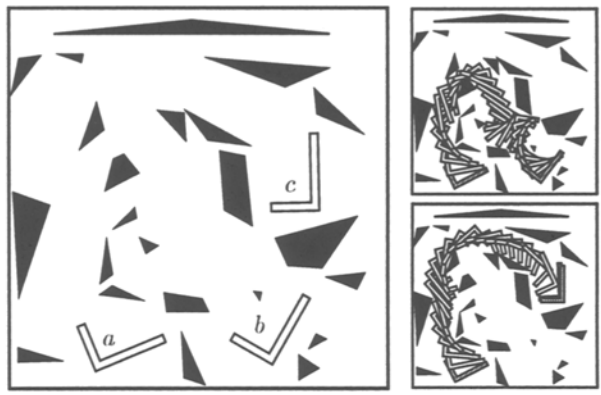
The values for the random walk parameters N_W and L_W are, respectively, 10 and 0.05. This guarantees that the time spent per query is bounded by approximately 0.3 seconds (on our machine). Clearly, if we allow more time per query, the method will be more successful in the query phase, and vice versa. Hence there is a trade-off between the construction time and the time allowed for a query.

In Figure 1 we have a free flying L-shaped robot, placed at the configurations a, b, and c. Simulation results are shown for the three corresponding queries, and two paths are shown, both smoothed in 1 second. We see that around 1 second of roadmap construction is required for obtaining roadmaps that solve the queries. These roadmaps consist of approximately 125 nodes.

In Figures 2 to 4 results are given for articulated robots.

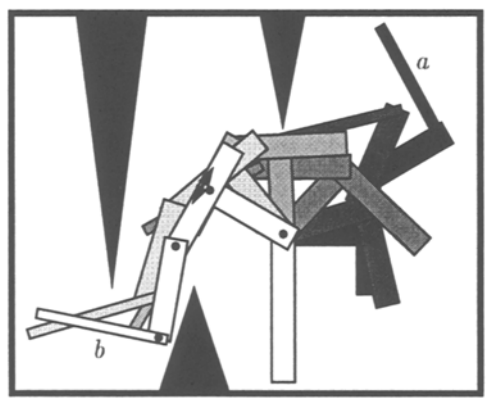
In the first two scenes, just one query is tested, and well the query (a, b). In both figures, several robot configurations along a path solving the query are displayed using various grey levels. The results of the experiments are given in the two tables. We see that the query in Figure 2 is solved in all cases after 10 seconds of construction time. Roadmap construction for 5 seconds however suffices to successfully answer the query in more than 90% of the cases. In Figure 3 we observe something similar. For both scenes the roadmaps constructed in 10 seconds contain around 500 nodes.

Figure 4 is a very difficult one. We have a seven dof robot in a very constrained environment. The configurations a, b, c, and d define 6 different queries, for which the results are shown. These where obtained by a customised implementation by Kavraki et al. [23]. In this implementation, optimised collision



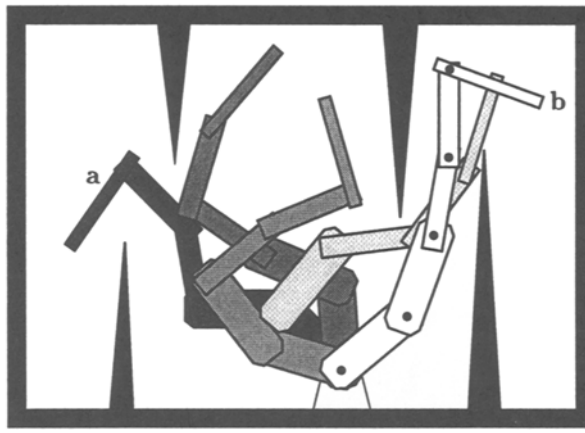
	0.25 sec.	0.5sec.	0.75 sec.	1.0 sec.	1.25 sec.
(a,b)	35%	55%	85%	95%	100%
(a,c)	20%	90%	100%	100%	100%
(b,c)	15%	75%	85%	100%	100%

Fig. 1. An L-shaped free-flying robot and its test configurations are shown. At the top right, we see two paths computed by the planner and smoothed in 1 second.



	2.5 sec.	5sec.	7.5sec.	10 sec.
(a,b)	53.3%	93.3%	100%	100%

Fig. 2. A four dof articulated robot, and a path.



	2.5 sec.	5sec.	7.5sec.	10 sec.
(a,b)	50%	87%	97%	100%

Fig. 3. A five dof articulated robot, and a path.

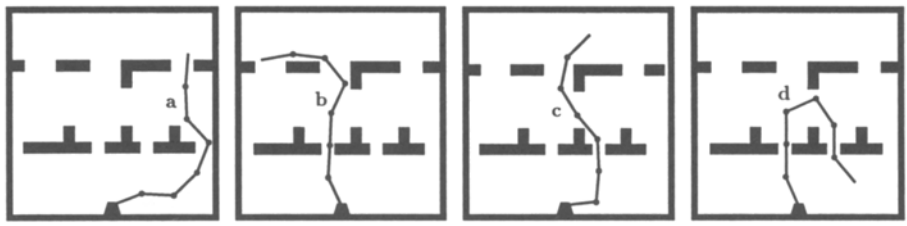
checking routines are used, as well as a robot-specific local planner. Furthermore, "difficult" nodes are heuristically identified during the roadmap construction phase, and "expanded" subsequently. We see that roughly 1 minute was sufficient to obtain roadmaps solving the 6 queries. These roadmaps consist of approximately 4000 nodes.

4 Application to nonholonomic robots

In this section we deal with nonholonomic mobile robots. More specifically, we apply *PPP* to car-like robots and tractor-trailer robots. We consider two types of car-like robots, i.e., such that can drive both forwards and backwards, and such that can only drive forwards. We refer to the former as *general car-like robots*, and to the latter as *forward car-like robots*. First however we give a brief overview on previous work on nonholonomic motion planning.

4.1 Some previous work on nonholonomic motion planning

Nonholonomic constraints add an extra level of difficulty to the path planning problem. The paths must (1) be collision free and (2) describe motions that are executable for the robot. We refer to such paths as *feasible* paths.



	20 sec.	30 sec.	40 sec.	50 sec.	60 sec.	70 sec.	80 sec.
(a,b)	25%	70%	80%	90%	100%	95%	100%
(a,c)	35%	55%	75%	90%	100%	100%	100%
(a,d)	15%	70%	80%	90%	95%	95%	100%
(b,c)	10%	40%	60%	80%	95%	100%	100%
(b,d)	5%	45%	65%	80%	95%	100%	100%
(c,d)	5%	40%	60%	80%	100%	95%	100%

Fig. 4. A seven dof articulated robot in a very constrained environment and the query test set.

For *locally controllable robots* [6], the existence of a feasible path between two configurations is equivalent to the existence of a collision free path, due to the fact that for any collision free path there exists a feasible path lying arbitrarily close to it. This fundamental property has led to a family of algorithms, decomposing the search in two phases. They first try to solve the geometric problem (i.e., the problem for the holonomic robot that is geometrically equivalent to the nonholonomic one). Then they use the obtained collision-free path to build a feasible one. So in the first phase the decision problem is solved, and only in the second phase are the nonholonomic constraints taken into account. One such approach was developed for car-like robots [26], using Reeds and Shepp works on optimal control to approximate the geometric path. In [34] Reeds and Shepp presented a finite family of paths composed of line segments and circle arcs containing a length-optimal path linking any two configurations (in absence of obstacles). The planner introduced in [26] replaces the collision-free geometric path by a sequence of Reeds and Shepp paths. This complete and fast planner was extended to the case of tractor-trailer robots, using near optimal paths numerically computed [27,12] (so far the exact optimal paths for the tractor-trailer system in absence of obstacle are unknown). The resulting planners are however neither complete nor time-efficient. The same scheme was used for systems that can be put into the *chained form.* For these systems, Tilbury et al. [50] proposed different controls to steer the system from

one configuration to another, in absence of obstacles. Sekhavat and Laumond prove in [38] that the *sinusoidal inputs* proposed by Tilbury et al. can be used in a complete algorithm transforming any collision-free path to a feasible one. This algorithm was implemented for a car-like robot towing one or two trailers, which can be put into the chained form, and finds paths in reasonable times [38]. A multi-level extension of this approach has been presented in [40] which further improves the running times of this scheme by separating the nonholonomic constraints mutually, and introducing separately. The scheme is however, as pointed out, only applicable to locally controllable robots. For example, forward car-like robots do not fall in this class.

Barraquand and Latombe [6] have proposed a heuristic brute-force approach to motion planning for nonholonomic robots. It consists of heuristically building and searching a graph whose nodes are small axis-parallel cells in C-space. Two such cells are connected in the graph if there exists a basic path between two particular configurations in the respective cells. The completeness of this algorithm is guaranteed up to appropriate choice of certain parameters, and it does not require local controllability of the robot. The main drawback of this planner is that when the heuristics fail it requires an exhaustive search in the discretised C-space. Furthermore, only the cell containing the goal configuration is reached, not the goal configuration itself. Hence the planner is inexact. Nevertheless, in many cases the method produces nice paths (with minimum number of reversals) for car-like robots and tractors pulling one trailer. For systems of higher dimension however it becomes too time consuming. Ferbach [11] builds on the approach of Barraquand and Latombe method in his *progressive constraints* algorithm in order to solve the problem in higher dimensions. First a geometric path is computed. Then the nonholonomic constraints are introduced progressively in an iterative algorithm. Each iteration consists of exploring a neighbourhood of the path computed in the previous iteration, searching for a path that satisfies more accurate constraints. Smooth collision-free paths in non-trivial environments were obtained with this method for car-like robots towing two and three trailers. The algorithm however does not satisfy any form of completeness.

The probabilistic path planner *PPP* has been applied to various types of nonholonomic robots. An advantage over the above single shot methods is the fact that a roadmap is constructed just ones, from which paths can subsequently be retrieved quasi-instantaneously. Also, local robot controllability is not required. A critical point of *PPP* when applied to nonholonomic robots is however the speed of the nonholonomic local planner. For car-like robots very fast local planners have been developed. Thanks to this, *PPP* applied to the car-like robots resulted in fast and probabilistically complete planners for car-like robots that move both forwards and backwards, as well as for such that can only move forwards [45,47]. Local planners for tractor-trailer robots

however tend to be much more time-consuming, which makes direct use of *PPP* less attractive. In [49] a local planner is presented and integrated into *PPP*, that uses exact closed form solutions for the kinematic parameters of a tractor-trailer robot. In [39] the local planner using sinusoidal inputs for chained form systems is used. For robots pulling more than one trailer, this local planner appeared to be too expensive for capturing the connectivity of the free C-space. For this reason, and inspired by the earlier mentioned works [26,27,12,38], in [39] a two-level scheme is proposed, where at the first level the portion of CS_{free} is reduced to a neighbourhood of a geometric path, and at the second level a (real) solution is searched for within this neighbourhood (by *PPP*). The multi-level algorithm proposed in [40] can in fact been seen as a generalisation of this two level scheme.

4.2 Description of the car-like and tractor-trailer robots

We model a car-like robot as a polygon moving in R^2, and its C-space is represented by $R^2 \times [0, 2\pi)$. The motions it can perform are subject to nonholonomic constraints. It can move forwards and backwards, and perform curves of a lower bounded turning radius r_{min}, as an ordinary car. A tractor-trailer robot is modelled as a car-like one, but with an extra polygon attached to it by a revolute joint. Its C-space is (hence) 4-dimensional, and can be represented by $R^2 \times [0, 2\pi) \times [-\alpha_{max}, \alpha_{max}]$, where α_{max} is the (symmetric) joint bound. The car-like part (the *tractor*) is a car-like robot. The extra part (the *trailer*) is subject to further nonholonomic constraints. Its motions are (physically) dictated by the motions of the tractor (For details, see for example [25,40]).

For car-like robots, the paths constructed will be sequences of translational paths (describing straight motions) and rotational paths (describing motions of constant non-zero curvature) only. It is a well-known fact [25] that if for a (general or forward) car-like robot a feasible path in the open free C-space exists between two configurations, then there also exists one that is a (finite) sequence of rotational paths. We include translational paths to enable straight motions of the robot, hence reducing the path lengths. For tractor-trailer robots we will use paths that are computed by transformation of the configuration coordinates to the chained form, and using sinusoidal inputs.

4.3 Application to general car-like robots

We now apply *PPP*, using an undirected graph, to general car-like robots. This again asks for filling in some of the (robot specific) details that have been left open in the discussion of the general method.

Filling in the details

The local planner: A *RTR path* is defined as the concatenation of a rotational path, a translational path, and another rotational path. Or, in other words, it is the concatenation of two circular arcs and a straight line segment, with the latter in the middle. The *RTR local planner* constructs the shortest *RTR* path connecting its argument configurations. Figure 5 shows two RTR paths. It can easily be proven that any pair of configurations is connected by a number of RTR paths (See [45] for more details). Furthermore, the RTR local planner satisfies a local topological property that guarantees probabilistic completeness (See Section 5).

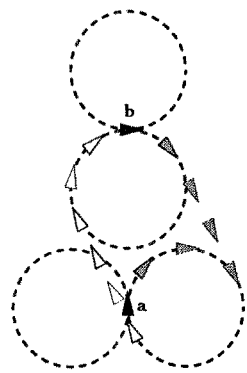


Fig. 5. Two RTR paths for a triangular car-like robot, connecting configurations a and b.

An alternative to the RTR local planner is to use a local planner that constructs the shortest (car-like) path connecting its argument configurations [34], [42]. Constructing shortest car-like paths is however a relatively expensive operation, and the construct requires more expensive intersection checking routines than does the RTR construct. On the other hand, RTR paths will, in general, be somewhat longer than the shortest paths, and, hence, they have a higher chance of intersection with the obstacles.

Collision checking for a RTR path can be done very efficiently, performing three intersection tests for translational and rotational sweep volumes. These sweep volumes are bounded by linear and circular segments (such objects are also called *generalised polygons*) and hence the intersection tests can be done exactly and efficiently. Moreover, the intersection tests for the

rotational path segments can be eliminated by storing some extra information in the graph nodes, hence reducing the collision check of a RTR path to one single intersection test for a polygon.

The distance measure: We use a distance measure that is induced by the RTR local planner, and can be regarded as an approximation of the (too expensive) induced "sweep volume metric", as described in Section 2.1. The distance between two configurations is defined as the length (in workspace) of the shortest RTR path connecting them. We refer to this distance measure as the *RTR distance measure*, and we denote it by D_{RTR}.

The random walks in the query phase: Random walks, respecting the car-like constraints, are required. The (maximal) number of these walks (per query) and their (maximal) lengths are parameters of the method, which we again denote by, respectively, N_W and L_W.

Let c_s be the start configuration of a random walk. As actual length l_W of the walk we take a random value from $[0, L_W]$. The random walk is now performed in the following way: First, the robot is placed at configuration c_s, and a random steering angle ψ and random velocity v are chosen. Then, the motion defined by (ψ, v) is performed until either a collision of the robot with an obstacle occurs, or the total length of the random walk has reached l_W. In the former case, a new random control is picked, and the process is repeated. In the latter case, the random walk ends.

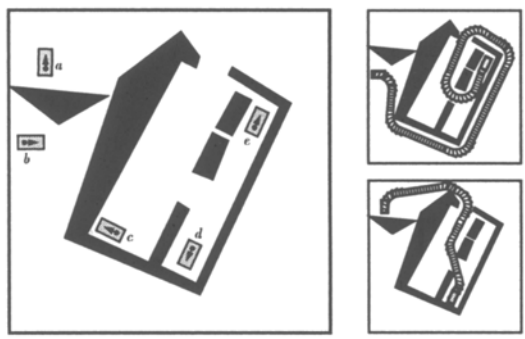
Good values for N_W and L_W must be experimentally derived (the values we use are given in the next section).

Node adding heuristics: We use the geometry of the workspace obstacles to identify areas in which is advantageous to add some extra, geometrically derived, non-random nodes. Particular obstacle edges and (convex) obstacle corners define such geometric nodes (See [47] for more details). Furthermore, as for free-flying robots, we use the (run-time) structure of the graph G in order to guide the node generation.

Simulation results We have implemented the planner as described above, and some simulation results are presented in this section. The planner was run on a machine as described in Section 3. Again the presented scenes correspond to the unit square with an obstacle boundary, and the chosen values for N_W and L_W are, respectively, 10 and 0.05. The simulation results are presented in the same form as for the holonomic robots in Section 3. That is, for different roadmap construction times we count how often graphs are obtained that solve particular, predefined, queries.

Figure 6 shows a relatively easy scene, together with the robot $\mathcal{A}$ positioned at a set of configurations $\{a, b, c, d, e\}$. The topology is simple and there are only a few narrow passages. We use $\{(a, b), (a, d), (b, e), (c, e), (d, e)\}$ as query test set T_Q. (At the top-right of Figure 6 paths solving the queries (a, d) and (b, e),

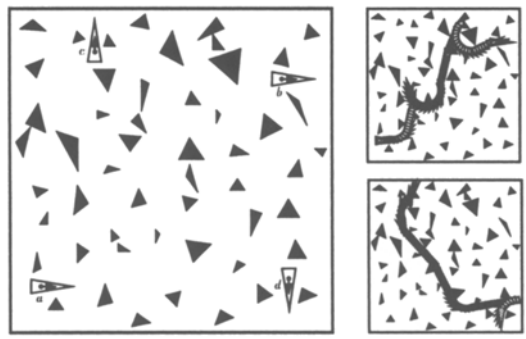
smoothed in 1 second, are shown.) The minimal turning radius r_{min} used in the experiments is 0.1, and the neighbourhood size *maxdist* is 0.5. We see that after only 0.3 seconds of roadmap construction, the networks solve each of the queries in most cases (but not all). Half a second of construction is sufficient for solving each of the queries, in all 20 trials. The corresponding roadmaps contain about 150 nodes.



	0.1 sec.	0.2 sec.	0.3 sec.	0.4 sec.	0.5 sec.
(a,b)	20%	90%	100%	100%	100%
(a,d)	35%	55%	85%	95%	100%
(b,e)	15%	75%	85%	100%	100%
(c,e)	60%	90%	100%	100%	100%
(d,e)	50%	85%	95%	100%	100%

Fig. 6. A simple scene. At the top right, two paths computed by the planner and smoothed in 1 second are shown.

Figure 7 (again together with a robot $\mathcal{A}$ placed at different configurations $\{a, b, c, d\}$), shows a completely different type of scene. It contains many (small) obstacles and is not at all "corridor-like". Although many individual path planning problems in this scene are quite simple, the topology of the free C-space is quite complicated, and can only be captured well with relatively complicated graphs. As query test set T_Q we use $\{(a, b), (a, c), (a, d), (c, d)\}$. Furthermore, as in the previous scene, $r_{min} = 0.1$ and $maxdist = 0.5$. Again, we show two (smoothed) paths computed by our planner (solving the queries (a, b) and (c, d)). We see that about 2 seconds of construction are required to obtain roadmaps that are (almost) guaranteed to solve each of the queries. Their number of nodes is about 350.



	0.25 sec.	0.5 sec.	0.75 sec.	1.0 sec.	1.5 sec.	2.0 sec.
(a,b)	30%	75%	95%	95%	100%	100%
(a,c)	20%	70%	85%	95%	100%	100%
(a,d)	50%	85%	95%	100%	100%	100%
(c,d)	35%	55%	75%	90%	90%	100%

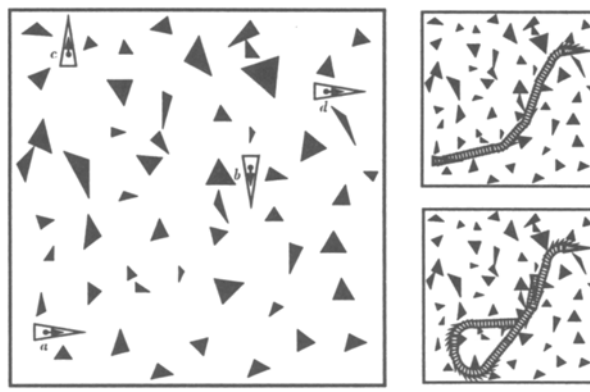
Fig. 7. A more complicated scene, and its test configurations. At the top right, two paths computed by the planner and smoothed in 1 second are shown.

4.4 Application to forward car-like robots

Forward car-like robots, as pointed out before, are not $\mathcal{C}$-symmetric. Hence, as explained in Section 2.3, directed instead of undirected graphs are used for storing the roadmaps. For details regarding the exact definition of the roadmap construction algorithm we refer to [32].

The robot specific components, such as the local planner, the metric, and the random walks are quite similar to those used for general car-like robots, as described in Section 4.3. The local planner constructs the shortest *forward RTR path* connecting its argument configurations. A forward RTR path is defined exactly as a normal RTR path, except that the rotational and translational paths are required to describe *forward* robot motions. The distance between two configurations is defined as the (workspace) length of the shortest forward RTR path connecting them. A random walk is performed as for general car-like robots, with the difference that the randomly picked velocity must be positive, and that, when collision occurs, the random walk is resumed from a random configuration on the previously followed trajectory (instead of from the configuration where collision occurred).

Simulation results In Figure 8 we give some results for the same scene as Figure 7. We see that the queries are most likely to be solved after 5 seconds of roadmap construction, and (almost) surely after 7.5 seconds, by roadmaps consisting of around 700 nodes. This means that about four times more time is required than for general car-like robots.



	1.0 sec.	2.0 sec.	3.0 sec.	4.0 sec.	5.0 sec.	7.5 sec.
(a, c)	35%	40%	75%	90%	90%	100%
(a, d)	30%	70%	85%	100%	100%	100%
(b, c)	15%	35%	55%	80%	95%	100%
(b, d)	20%	45%	45%	70%	90%	100%

Fig. 8. Motion planning for a forward car-like robot.

4.5 Application to tractor-trailer robots

As last example of nonholonomic robots, we now (briefly) consider tractor-trailer robots, and well such that can drive both forwards and backwards. These robots have symmetrical control systems and, hence, undirected underlying graphs are sufficient. We will not go into many details. We refer to [39,40] for a more thorough discussion of the topic. We use a local planner, by Sekhavat and Laumond [38], that transforms its configuration coordinates into the chained form, and uses sinusoidal inputs. We refer to it as the *sinusoidal local planner*. This local planner verifies a local topological property that guarantees probabilistic completeness of the global planner. As distance measure we use

(cheap) approximations of the workspace lengths of the local paths. The random walks in the query phase are basically as those for general car-like robots, except that the trailers orientation must be kept track of during each motion of the tractor. This can be done using exact closed form solutions for the kinematic parameters of tractor-trailer robots under constant curvature motions of the tractor [49]. If, during such a motion, the tractors orientation gets out of bounds (relative to the orientation of the tractor), this is treated as a collision.

Simulation results See Figure 9 for two feasible paths computed by the Probabilistic Path Planner. The computation time of the roadmap from which the paths where retrieved took about 10 seconds (on the average).

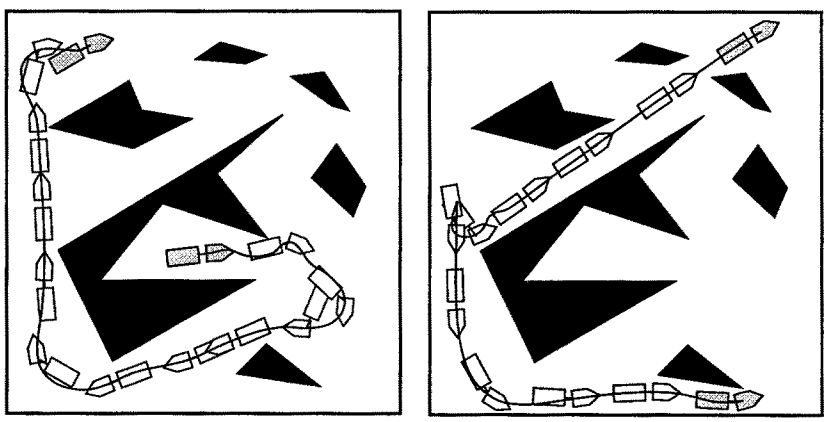


Fig. 9. Two feasible paths for a tractor-trailer robot, obtained in 10 seconds.

5 On probabilistic completeness of probabilistic path planning

In this section we discuss some aspects regarding *probabilistic completeness* of *PPP*, and we prove this completeness for the specific planners described in this chapter. We will assume a slightly simplified version of the planning scheme. Instead of trying to connect the start configuration s and goal configuration g to the graph with some graph retractions (as described in Section 2.2), we simply add s and g to the graph as (initial) nodes. A query consists of just a

graph search. This simplification of the query phase is for ease of presentation. All results presented in this section directly hold for the case where queries are performed as described earlier (in Section 2.2), using graph retractions and random walks.

A path planner is called *probabilistically complete* if, given any problem that is solvable in the open free C-space, the probability that the planner solves the problem goes to 1 as the running time goes to infinity. Hence, a probabilistically complete path planner is guaranteed to solve such a problem, provided that it is executed for a sufficient amount of time. For ease of presentation we introduce some shorthand notations. We denote the version of *PPP* using undirected underlying graphs (respectively directed graphs) by PPP_u (respectively PPP_d). The notation $PPP_u(L)$ (respectively $PPP_d(L)$) is used for referring to PPP_u (respectively PPP_d) with a specific local planner L. We say L is *symmetric* iff, for arbitrary configurations a and b, $L(a,b)$ equals $L(b,a)$ reversed. Furthermore, we again denote the C-space, respectively the free C-space, by $\mathcal{C}$, respectively $\mathcal{C}_f$.

We point out that with *PPP* one obtains a probabilistically complete planner for any robot that is *locally controllable* (see below), provided that one defines the local planner properly. If, in addition to the local controllability, the robot also has a symmetric control system then $PPP_u(L)$ is suitable, otherwise $PPP_d(L)$ must be used. In Section 5.1 we define a general property for local planners that is sufficient for probabilistic completeness of *PPP*, and we point out that, given the local controllability of the robot, a local planner satisfying this property always exists (but it must be found). We also mention a relaxation of the property, that guarantees probabilistic completeness of $PPP_u(L)$ as well, for locally controllable robots with symmetric control systems. All holonomic robots, as well as for example general car-like robots and tractor-trailer robots, fall into this class. Forward car-like robots however are not locally controllable (and neither symmetric). In Section 5.2 we show that all the planners described in this chapter are probabilistically complete.

First we define the concept local controllability (in the literature also referred to as small-time local controllability or local-local controllability), adopting the terminology introduced by Sussman [43]. Given a robot $\mathcal{A}$, let $\Sigma_{\mathcal{A}}$ be its control system. That is, $\Sigma_{\mathcal{A}}$ describes the velocities that $\mathcal{A}$ can attain in C-space. For a configuration c of a robot $\mathcal{A}$, the set of configurations that $\mathcal{A}$ can reach within time T is denoted by $A_{\Sigma_{\mathcal{A}}}(\leq T, c)$. $\mathcal{A}$ is defined to be *locally controllable* iff for any configuration $c \in \mathcal{C}$ $A_{\Sigma_{\mathcal{A}}}(\leq T, c)$ contains a neighbourhood of c (or, equivalently, a ball centred at c) for all $T > 0$. It is a well-known fact that, given a configuration c, the area a locally controllable robot $\mathcal{A}$ can reach without leaving the ϵ-ball around c (for any $\epsilon > 0$) is the entire open ϵ-ball around c.

5.1 The general local topology property

We assume now that robot $\mathcal{A}$ is locally controllable. For probabilistic completeness of *PPP* a local planner L is required that exploits the local controllability of $\mathcal{A}$. This will be the case if L has what we call the *general local topological property*, or *GLT-property*, as defined in Definition 6 using the notion of *ϵ-reachability* introduced in Definition 5. We denote the ball (in C-space) of radius ϵ centred at configuration c by $B_\epsilon(c)$.

Definition 5. *Let L be a local planner for $\mathcal{A}$. Furthermore let $\epsilon > 0$ and $c \in \mathcal{C}$ be given. The* ϵ-reachable area of c by L, *denoted by $R_{L,\epsilon}(c)$, is defined by*

$$R_{L,\epsilon}(c) = \{\tilde{c} \in B_\epsilon(c) | L(c,\tilde{c}) \text{ is entirely contained in } B_\epsilon(c)\}$$

Definition 6. *Let L be a local planner for $\mathcal{A}$. We say L has the GLT-property iff*

$$\forall \epsilon > 0 : \exists \delta > 0 : \forall c \in \mathcal{C} : B_\delta(c) \subset R_{L,\epsilon}(c)$$

We refer to $B_\delta(c)$ as the ϵ-reachable δ-ball *of c.*

A local planner verifying the GLT-property, at least in theory, always exists, due to the robots local controllability. Theorem 5.1 now states that this property is sufficient to guarantee probabilistic completeness of *PPP*. That is, of $PPP_u(L)$ if L is symmetric, and of $PPP_d(L)$ otherwise.

Theorem 5.1. *If L is a local planner verifying the GLT-property, then* PPP(L) *is probabilistically complete.*

Proof. The theorem can be proven quite straightforwardly (for both $PPP_u(L)$ and $PPP_d(L)$). Assume L verifies the GLT-property. Given two configurations s and g, lying in the same connected component of the open free C-space, take a path P that connects s and g and lies in the open free C-space as well. Let ϵ be the C-space clearance of P (that is, the minimal distance between P and a C-space obstacle), and take $\delta > 0$ such that $\forall c \in \mathcal{C} : B_\delta(c) \subset R_{L,\frac{3}{4}\epsilon}(c)$. Then, consider a covering of P by balls $B_1, \ldots, B_k$ of radius $\frac{1}{4}\delta$, such that balls B_i and B_{i+1}, for $i \in \{1, \ldots, k-1\}$, partially overlap. Assume each such ball B_i contains a node v_i of G. Then, $|v_i - v_{i+1}| \leq \delta$, and each node v_i has a C-space clearance of at least $\epsilon - \frac{1}{4}\delta \geq \frac{3}{4}\epsilon$ (since $\delta \leq \epsilon$). Hence, due to the definition of δ, we have

$$L(v_i, v_{i+1}) \subset B_{\frac{3}{4}\epsilon}(v_i) \subset \mathcal{C}_f$$

It follows that if all the balls $B_1, \ldots, B_k$ contain a node of G, s and g will be graph-connected. Since, due to the random node adding, this is guaranteed to be the case within a finite amount of time, this concludes the proof. See also Figure 10. ∎

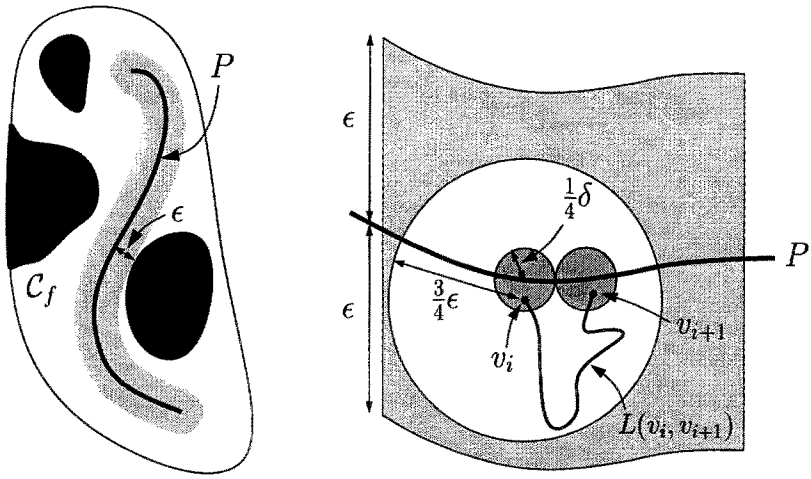


Fig. 10. Path P has clearance $\epsilon > 0$. If $\delta > 0$ is chosen such that $\forall c \in \mathcal{C} : B_\delta(c) \subset R_{L,\frac{3}{4}\epsilon}(c)$, then we see that nodes in overlapping $\frac{1}{4}\delta$-balls, centred at configurations of P, can always be connected by the local planner.

Clearly, given a locally controllable robot, the GLT-property is a proper criterion for choosing the local planner (sufficient conditions for local controllability of a robot are given in, e.g., [44]). Path planning among obstacles for car-like robots using local planners with the GLT-property has also been studied by Laumond [28,18].

For locally controllable robots with symmetric control systems, a weaker property exists that guarantees probabilistic completeness as well. We refer to this property as the *LTP-property*. The basic relaxation is that we no longer require the ϵ-reachable δ-ball of a configuration a to be centred around c. We do however make a certain requirement regarding the relationship between configurations and the corresponding ϵ-reachable δ-balls. Namely, it must be described by a *Lipschitz continuous function*. For a formal definition of the LTP-property and a proof of probabilistic completeness with local planners verifying it, we refer to [47].

5.2 Probabilistic completeness with the used local planners

The local planners used for holonomic robots, general car-like robots, forward car-like robots, and tractor-trailer robots, as described earlier in this chapter, guarantee probabilistic completeness.

Locally controllable robots The general holonomic local planner L for holonomic robots constructs the straight line path (in C-space) connecting its argument configurations. It immediately follows that $R_{\epsilon,L}(c) = B_\epsilon(c)$, for any configuration c and any $\epsilon > 0$. Hence, L verifies the GLT-property.

Theorem 5.2. $\mathrm{PPP}_u(L)$, *with L being the general holonomic local planner, is probabilistically complete for all holonomic robots.*

The planner for general car-like robots uses the RTR local planner. One can prove that this planner verifies the LTP-property [47]. Again, as stated in the following theorem, this guarantees probabilistic completeness.

Theorem 5.3. $\mathrm{PPP}_u(L)$, *with L being the RTR local planner, is probabilistically complete for general car-like robots.*

Regarding tractor-trailer robots, Sekhavat and Laumond prove in [38] that the sinusoidal local planner, used for the tractor-trailer robots, verifies the GLT-property. Hence, for tractor-trailer robots we also have probabilistic completeness.

Theorem 5.4. $\mathrm{PPP}_u(L)$, *with L being the sinusoidal local planner, is probabilistically complete for tractor-trailer robots (with arbitrary number of trailers).*

Forward car-like robots As pointed out before, the theory of the previous sections applies only to robots that are locally controllable. If a robot does not have this property, a local planner verifying the GLT-property will not exist. A local planner verifying the weaker LTP-property may exist, but this planner will not be symmetric (this would imply the existence of a local planner verifying GTP).

Forward car-like robots are not locally controllable. One can nevertheless prove probabilistic completeness of $PPP_d(L)$, with L being the RTR forward local planner. That is, one can prove that, given two configurations s and g such that there exists a feasible path in the open free C-space connecting them, $PPP_d(L)$ will surely solve the problem within finite time. The proof, which does not directly generalise to other cases, uses a property of RTR forward paths stated in Lemma 5.5.

Lemma 5.5. *Let L be the RTR forward local planner, and let Q be a RTR forward path connecting configurations a and b with a straight line path of non-zero length and both arc paths of total curvature less than $\frac{1}{2}\pi$. Then:*

$\forall \epsilon > 0 : \exists \delta > 0 : \forall (\tilde{a}, \tilde{b}) \in B_\delta(a) \times B_\delta(b) : L(\tilde{a}, \tilde{b})$ *lies within distance ϵ of* Q [1]

[1] We say a path P lies within distance ϵ of a path Q, iff $\forall p \in P : \exists q \in Q : |p - q| \leq \epsilon$ (in C-space)

Theorem 5.6. $\mathrm{PPP}_d(L)$, *with* L *being the RTR forward local planner, is probabilistically complete for forward car-like robots.*

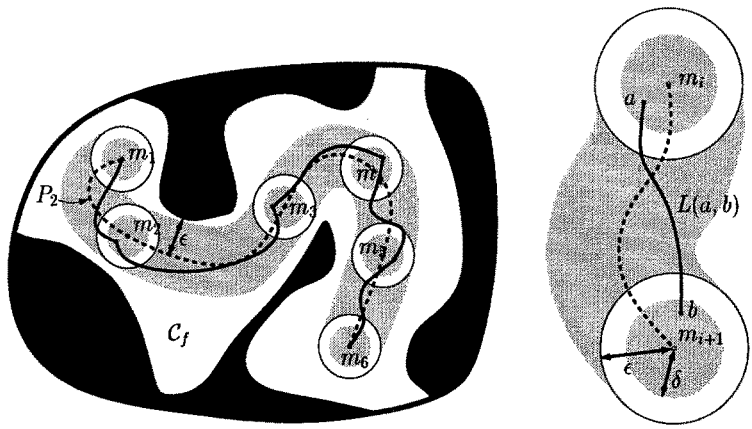


Fig. 11. This figure illustrates the proof of Theorem 5.6. P_2 is a path, feasible for a forward car-like robot, of clearance $\epsilon > 0$. Centred at the configurations m_i are balls B_i of a radius $\delta > 0$, such that any pair of configurations $(a, b) \in B_i \times B_{i+1}$ is connected by the RTR forward local planner L with a path lying within distance ϵ of P_2, and hence lying in C_f.

We give only a sketch of the proof here (See also Figure 11). Let L be the RTR forward local planner. Assume P_1 is a path in the open free C-space connecting a (start) configuration s to a (goal) configuration g, that is feasible for our forward car-like robot $\mathcal{A}$. Then, one can prove, there exists also a feasible path P_2 in the open free C-space, connecting s to g, that consists of (a finite number of) straight line segments and circular arcs, such that no two distinct arcs are adjacent and each arc has a total curvature of less than $\frac{1}{2}\pi$.[2]

Assume k is the number of arcs in P_2. Let $m_1 = s$, $m_k = g$, and $\{m_2, \ldots, m_{k-1}\}$ be points on P_2 such that m_i is the midpoint of the i-th arc of P_2 (that is, the unique point on the arc with equal distance to both end-points). Clearly, m_i is connected m_{i+1} by a forward RTR path with a straight line segment of non-zero length and both arc paths of total curvature less than $\frac{1}{2}\pi$ (for all $j \in \{1, \ldots, k-1\}$).

[2] This does not necessarily hold if P_1 consists of just one or two circular arcs of maximal curvature. In this case however P_1 can be found directly with the local planner.

Let $\epsilon > 0$ be the clearance of P_2, and take $\delta > 0$ such that, for all $j \in \{1, \ldots, k-1\}$:

$$\forall (a, b) \in B_\delta(m_j) \times B_\delta(m_{j+1}) : L(a, b) \text{ lies within distance } \epsilon \text{ of } Q$$

It follows from Lemma 5.5 that such a $\delta > 0$ always exists. When a node of G is present in every ball $B_\delta(m_j)$ for $2 \leq j < k$, G will contain a path connecting s to g. We know, due to the probabilistic nature of the node adding, that the probability of obtaining such a graph grows to 1 when the roadmap construction time goes to infinity.

6 On the expected complexity of probabilistic path planning

In the previous section we have formulated properties of local planners that guarantee probabilistic completeness of *PPP* for locally controllable robots. If these properties are satisfied, we know that as the running time of *PPP* goes to infinity, the probability of solving any solvable problem goes to 1. However, nothing formal has yet been said about (expected) convergence times of the algorithm. In practice, one will not be satisfied with the guarantee that "eventually a path will be found". For real life applications, some estimate of the running time beforehand is desirable.

Simulation results obtained by the application of *PPP* on certain "typical" problems can increase our trust in the planners performance and robustness, but they do not describe a formal relation between the probabilities of failure and running times in general, and neither do they provide a theoretical explanation for the empirically observed success of the probabilistic planner. Recently some first theoretical results on expected running times of probabilistic planners have been obtained.

Kavraki et al. [22,20,3] show that, under certain geometric assumptions about the free C-space C_f, it is possible to establish a relation between the probability that probabilistic planners like *PPP*[3] find paths solving particular problems, and their running times. They suggest two such assumptions, i.e., the *visibility volume assumption* and the *path clearance assumption*. We will discuss these assumptions and present the main results obtained by Kavraki et al.. Also, we will discuss to what extent these results hold for nonholonomic robots, and, where possible, we will adapt them appropriately. Furthermore, we introduce a new assumption on configuration space, the *ϵ-complexity assumption*, under which it is possible to relate the success probabilities and running times of *PPP* to the complexity of the problems that are to be solved.

[3] Kavraki et al. refer to *PPP* by the name *PRM* (Probabilistic RoadMap planner).

Throughout this section we use the notations introduced in the previous section, and, unless stated otherwise, we will assume that *PPP* with undirected underlying graphs (i.e., PPP_u) is used.

6.1 The visibility volume assumption

The visibility volume assumption uses a notion of "visibility" defined by the used local planner. A free configuration a is said to "see" a free configuration b if the local path $L(a, b)$ lies entirely in $\mathcal{C}_f$. The visibility volume assumption now states that every free configuration "sees" a subset of $\mathcal{C}_f$ whose volume is at least an ϵ fraction of the total volume of $\mathcal{C}_f$. If this holds, $\mathcal{C}_f$ is called *ϵ-good.*

The analyses assumes a somewhat more complex planner than *PPP* as defined in Section 2. It differs from *PPP* in that after a probabilistic roadmap $G = (V, E)$ has been constructed (by the roadmap construction algorithm in Section 2.1), an extra post-processing step is performed, referred to by the authors as *Permeation.* Permeation assumes the existence of a complete planner, that is, a planner solving any solvable problem, and returning failure for any non-solvable one. What permeation does, is invoking the complete planner for certain pairs $(a, b) \in V \times V$ that are not graph connected. Planners based on this scheme have been implemented by Kavraki et al. (E.g., [21]). However, instead of a complete planner (which, in general, is not available) the potential field planner *RPP* has been utilised. Since *RPP* is only probabilistically complete, the mentioned planners are merely approximations of the algorithm sketched above.

Due to the assumed completeness of the invoked planner, provided that the complete planner is invoked for enough pairs of nodes in V, permeation leads to a roadmap where every connected component of $\mathcal{C}_f$ contains at most one connected component of the roadmap G. For convenience, we will say that such roadmaps have *perfect connectivity.*

Let us now assume that G_P has such perfect connectivity. Then, if both s and g "see" a node of G_P, the planner will return a path solving this problem if it is solvable, and return failure otherwise. In other words, on the portion of $\mathcal{C}_f$ that "sees" some node of the roadmap, the planner is complete. Note that during the permeation step, no nodes are added to G. Hence, the question whether a solvable query will be answered correctly depends solely on the set of nodes V in G. Theorem 6.1 addresses this question. V is called *adequate* if the portion of $\mathcal{C}_f$, not visible from any $c \in V$, has a volume of at most $\frac{1}{2}\epsilon \cdot \text{Volume}(\mathcal{C}_f)$. The theorem gives a lower bound for the probability of V being adequate.

Theorem 6.1. ***(Kavraki et al. [22], Barraquand et al. [3])*** *Assume $G = (V, E)$ is a roadmap generated by* PPP *in a free C-space that is ϵ-good. Let*

$\beta \in (0,1]$ be a real constant, and let C be a positive constant large enough such that $\forall x \in (0,1] : (1-x)^{(\frac{C}{x}\log\frac{1}{x})} \leq x\frac{\beta}{4}$ Now, if $|V| \geq \frac{C}{\epsilon}\log\frac{1}{\epsilon}$, then V is adequate with probability at least $1-\beta$.

Regarding the complexity of the roadmap construction, one must estimate the number of calls to the complete planner during the permeation step, required for obtaining a roadmap G_P of perfect connectivity. Theorem 6.2 provides such an estimate.

Theorem 6.2. ***(Kavraki et al. [22], Barraquand et al. [3])*** *Let $w_1 \geq w_2 \geq \cdots \geq w_k$ be the sizes of the connected components of the roadmap $G = (V, E)$. There exists a (randomised) algorithm that extends G to a roadmap G_P of perfect connectivity, whose expected number of calls of the complete planner is at most:*

$$2\sum_{i=1}^{k} i \cdot w_i - |V| - k$$

Furthermore, with high probability, the number of calls is at most:

$$O\left(\sum_{i=1}^{k} i \cdot w_i \cdot \log(|V|)\right)$$

Symmetry of the local planner L is required and assumed in the above. However, apart from this, the ϵ-goodness assumption is very general, and the given analysis is not only valid for holonomic robots (on which the authors focus), but also for nonholonomic ones.

However, the question arises in how far the theoretical results are "practical" for nonholonomic robots, since the "visibility" induced by a nonholonomic local planner will be a quite hard to grasp concept that can hardly be regarded as a geometric property of $\mathcal{C}_f$. There does not seem to be much hope that we will ever be able to measure the ϵ-goodness of $\mathcal{C}_f$ for nonholonomic robots, in non-trivial cases (insofar as there is such hope for holonomic ones).

6.2 The path clearance assumption

In the path clearance assumption, there exists a collision-free path $\mathcal{P}$ between the start configuration s and goal configuration g, that has some clearance $\epsilon > 0$ with the C-space obstacles. Throughout this section we denote the volume of an object A by $\mathcal{V}(A)$. In [20], Kavraki et al. study the dependence of the failure probability of *PPP* (the normal version) to connect s and g on (1) the length of $\mathcal{P}$, (2) the clearance ϵ, and (3) the number of nodes in the probabilistic roadmap G. Their main result is described by the following theorem:

Theorem 6.3. ***(Kavraki et al. [20], Barraquand et al. [3])*** *Let $\mathcal{A}$ be a holonomic robot, L be the general holonomic local planner (for $\mathcal{A}$), and $G = (V, E)$ be a graph constructed by* PPP(L). *Assume configurations s and g are connectable by a path $\mathcal{P}$ of length λ, that has a clearance $\epsilon > 0$ with the (C-space) obstacles. Let $\alpha \in (0, 1]$ be a real constant, and let a be the constant $\frac{1}{2^n}\mathcal{V}(\mathcal{B}_1)/\mathcal{V}(\mathcal{C}_f)$, where $\mathcal{B}_1$ denotes the unit ball in the C-space $\mathbb{R}^n$. Now if $|V|$ is such that*

$$\frac{2\lambda}{\epsilon}(1 - a\epsilon^n)^{|V|} \leq \alpha$$

then, with probability at least $1 - \alpha$, s and g will be graph-connected in G (See also Figure 12).

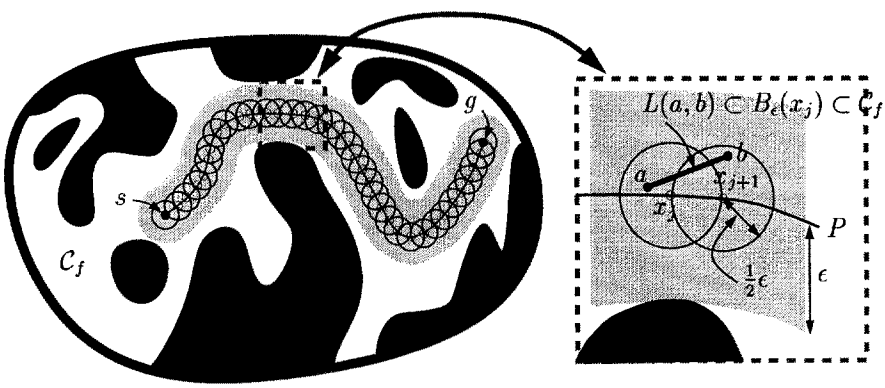


Fig. 12. We see that configuration s is connectable to configuration g by a P of clearance ϵ. Let $x_0 = s, x_1, \ldots, x_k = g$ be points on P, such that $|x_j - x_{j+1}| \leq \frac{1}{2}\epsilon$, for all j. If each ball $B_{\frac{1}{2}\epsilon}(x_j)$ contains a node of G, then s and g will be graph-connected.

The proof of this theorem is quite straightforward. Given a path P of clearance $\epsilon > 0$, one can consider a covering of P by balls of radius $\frac{1}{2}\epsilon$ as shown in Figure 12, and bound the probability that one of these balls contains no node of G. Since, if each of these balls does contain a node, G is guaranteed to contain a path connecting the start and goal configuration, this gives an upper bound for the failure probability of *PPP*.

A number of important facts are implied by Theorem 6.3. E.g., the number of nodes required to be generated, in order for the planner to succeed with probability at least $1 - \alpha$, is logarithmic in $\frac{1}{\alpha}$ and λ, and polynomial in $\frac{1}{\epsilon}$. Furthermore, the failure probability α is linear in the path length λ.

The analyses assumes the use of the general holonomic local planner (as described in Section 3.1). Hence it is assumed that the robot is holonomic. An underlying assumption is namely that the ϵ-reachable area of any configuration c consists of the entire ϵ-ball $B_\epsilon(c)$, surrounding c. From theoretical point of view, as pointed out in the previous section, for any locally controllable robot a local planner exists for which the ϵ-reachable area of any configuration c consists of the entire open ϵ-ball centred at c. Such a local planner would allow for the result assuming the path clearance to be directly applied to such robots. However, it is not realistic to assume the "ϵ-ball reachability" for a nonholonomic local planner, since for most robots we are not able to construct such local planners, and, if we could, they would probably be vastly outperformed (in terms of computation time) by simple local planners verifying only weaker (but sufficient) topological properties, such as those presented in the previous section. However, the analyses presented in [20] can be extended to the case where the local planner verifies only the GLT-property. Through this, we can give running time estimates for locally controllable nonholonomic robots that are realistic in the sense that we can actually build the planners that we analyse. Corollary 6.4 extends the result of Theorem 6.3 to locally controllable nonholonomic robots with local planners verifying the *GLT*-property.

Corollary 6.4. *Let $\mathcal{A}$ be a fully controllable robot, L be a local planner for $\mathcal{A}$ verifying the* GLT*-property, and $G = (V, E)$ be a graph constructed by* PPP(L). *Assume configurations s and g are connectable by a path $\mathcal{P}$ of length λ, that has a clearance $\epsilon > 0$ with the (configuration space) obstacles. Take $\delta > 0$ such that*

$$\forall c \in \mathcal{C} : B_\delta(c) \subset R_{L,\frac{3}{4}\epsilon}(c)$$

Let $\alpha \in (0, 1]$ be a real constant, and $\mathcal{V}_1$ be the volume of the unit ball in the C-space $\mathbb{R}^n$. Now if $|V|$ is such that

$$\frac{2\lambda}{\delta}\left(1 - \frac{\mathcal{V}_1}{4^n \mathcal{V}(\mathcal{C}_f)}\delta^n\right)^{|V|} \leq \alpha$$

then, with probability at least $1 - \alpha$, s and g will be graph-connected in G.

Since, by definition of the *GLT*-property, δ is a constant with respect to ϵ, the dependencies implied by Theorem 6.3 hold for nonholonomic robots as well.

6.3 The ϵ-complexity assumption

A drawback of Theorem 6.3 and the above corollary is that no relation is established between the failure probability and the *complexity* of a particular

problem. In our opinion, to a considerable extent, the observed success of *PPP* lies in the fact that not the complexity of the C-space, but the complexity of the resulting path defines the (expected) running time of *PPP*. For example, assume a particular problem is solvable by a path P of clearance $\epsilon > 0$, consisting of say 4 straight line segments. Consider three balls of radius ϵ, centred at the 3 inner nodes of P. Then, as is illustrated in Figure 13, it suffices that all the 3 balls contain a node of G to guarantee that the problem is solved. We see in this example that the failure probability in no way relates to the length of the path, and neither to the complexity of C_f. The only relevant factors are the clearance and the complexity of the path. Definition 7 introduces the notion of *ϵ-complexity*, which captures this measure of problem complexity. We refer here to a path composed of k straight line segments as a piecewise linear path of complexity k.

Definition 7. *Given a holonomic robot and a particular path planning problem (s, g), let P be the lowest complexity piecewise-linear path connecting s and g, that has a C-space clearance of $\epsilon > 0$. We define the* ϵ-complexity *of problem (s, g) as the complexity of P.*

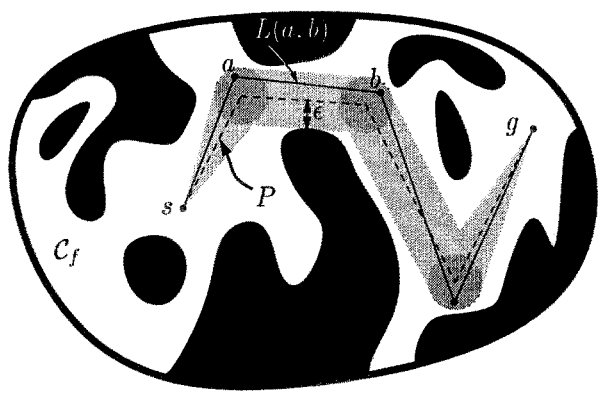


Fig. 13. We see that configuration s is connectable to configuration g by a piecewise linear path P (dashed) of complexity 4 and clearance ϵ. If each of 3 dark grey balls (of radius ϵ, placed at the vertices of P) contains a node of G, then the G contains a path, lying in the grey area, that connects s and g.

Theorem 6.5 gives a result relating the failure probability of *PPP* to the *ϵ-complexity* of the problem to be solved. It applies only to holonomic robots and assumes the use of the general holonomic local planner.

Theorem 6.5. *Let $\mathcal{A}$ be a holonomic robot, L be the general holonomic local planner (for $\mathcal{A}$), and $G = (V, E)$ be a graph constructed by* PPP(L). *Assume (s, g) is a problem of ϵ-complexity ζ. Let $\alpha \in (0, 1]$ be a real constant, and $\mathcal{V}_1$ be the volume of the unit ball in the C-space $\mathbb{R}^n$. Now if $|V|$ is such that*

$$(\zeta - 1)\left(1 - \frac{\mathcal{V}_1}{\mathcal{V}(\mathcal{C}_f)}\epsilon^n\right)^{|V|} \leq \alpha$$

then, with probability at least $1 - \alpha$, s and g will be graph-connected in G.

This theorem can be proven quite easily. Given a problem of ϵ-complexity ζ, there exists a piecewise linear path P of complexity ζ and clearance $\epsilon > 0$ solving it. We can place balls of radius ϵ at the vertices of P, and bound the probability the one of these balls contains no node of G. Since, if each of these balls does contain a node, G is guaranteed to contain a path connecting the start and goal configuration, this gives us an upper bound for the failure probability of *PPP*.

So we now also have a linear dependence of the failure probability, and a logarithmic dependence of $|V|$, on the *complexity* ζ of the path P, that is, on the ϵ-complexity of the problem. We note that the existence of a path of a certain clearance $\epsilon > 0$ and implies the existence of a piecewise linear path of a similar clearance.

7 A multi-robot extension

We conclude this chapter with an extension of *PPP* for solving multi-robot path planning problems. That is, problems involving a number of robots, present in the same workspace, that are to change their positions while avoiding (mutual) collisions. Important contributions on multi-robot path planning include [37,13,10,51,41,8,30,29,4,5,17,2,52]. For overviews we refer to [25] and [15].

Most previous successful planners fall into the class of *decoupled planners*, that is, planners that first plan separate paths for the individual robots more or less independently, and only in a later stage, in case of collisions, try to adapt the paths locally to prevent the collisions. This however inherently leads to planners that are not complete, that is, that can lead to deadlocks. To obtain some form of completeness, one must consider the separate robots as one composite system, and perform the planning for this entire system. However, this tends to be very expensive, since the composite C-space is typically of high dimension, and the constraints of all separate add up.

For example, multi-robot problems could be tackled by direct application of *PPP*. The robot considered would be composed of the separate "simple" robots, and the local planner would construct paths for this composite robot.

This is a very simple way of obtaining (probabilistically complete) multi-robot planners. However, as mentioned above, a drawback is the high dimension of the configuration space, which, in non-trivial scenes, will force *PPP* to construct very large roadmaps for capturing the structure of $\mathbf{C}_f$. Moreover, each local path in such a roadmap will consists of a number of local paths for the simple robots, causing the collision checking to be rather expensive.

In this section we describe a scheme where a roadmap for the composite robot is constructed only after a discretisation step that allows for disregarding the actual C-space of the composite robot. See Figure 14 for an example of a multi-robot path planning problem, and a solution to it, computed by a planner based on the scheme.

We will refer to the separate robots $\mathcal{A}_1, \ldots, \mathcal{A}_n$ as the *simple robots.* One can also consider the simple robots together to be one robot (with many degrees of freedom), the so-called *composite robot.* A feasible path for the composite robot will be referred to as a *coordinated path.* We assume in this paper that the simple robots are identical, although, with minor adaptions, the presented concepts are applicable to problems involving non-identical robots as well.

A roadmap for the composite robot is constructed in two steps. First, a *simple roadmap* is constructed for just one robot with *PPP.* Then n of such roadmaps are combined into a roadmap for the composite robot (consisting of n simple robots). We will refer to such a composite roadmap as a *super-graph.* After such a super-graph has been constructed, which needs to be done just once for a given static environment, it can be used for retrieval of coordinated paths. We will present two super-graph structures: *flat super-graphs* and *multi-level super-graphs.* The latter are a generalisation of flat super-graphs, that consume much less memory for problems involving more than 3 robots.

The scheme is a flexible one, in the sense that it is easily applicable to various robot types, provided that one is able to construct simple roadmaps for one such robot. Furthermore, proper construction of the simple roadmaps guarantees *probabilistic completeness* of the resulting multi-robot planners [46]. In this paper we apply the super-graph approach to *car-like robots.* We give simulation results for problems involving up to 5 robots moving in the same constrained environment.

7.1 Discretisation of the multi-robot planning problem

The first step of our multi-robot planning scheme consists of computing a simple roadmap, that is, a roadmap for the simple robot $\mathcal{A}$. We assume that this roadmap is stored as a graph $G = (V, E)$, with the nodes V corresponding to collision-free configurations, and the edges E to feasible paths, also referred to as *local paths.* We say a node *blocks* a local path, if the volume occupied by $\mathcal{A}$

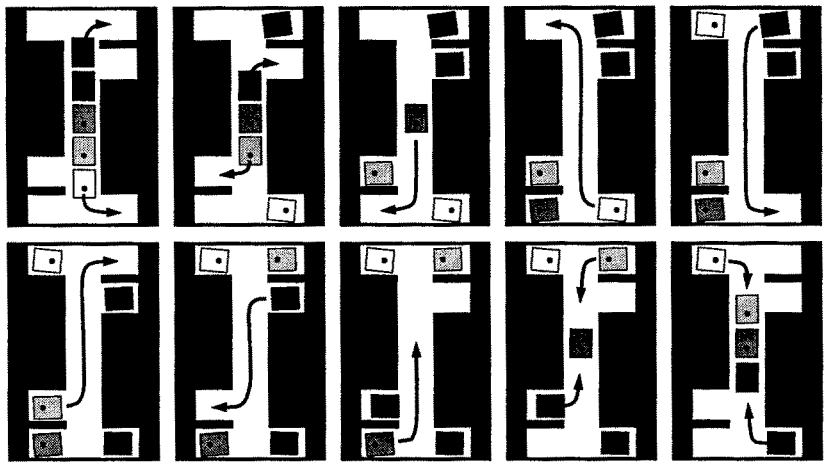


Fig. 14. An example of a multi-robot path planning problem, with a solution shown (generated by the multi-level super-graph planner). Five car-like robots are in a narrow corridor, and they are to reverse their order.

when placed at the node intersects the volume swept by $\mathcal{A}$ when moving along the local path. Basically, any algorithm that constructs roadmaps can be used in this phase. We will use *PPP*.

Given a graph $G = (V, E)$ storing a simple roadmap for robot $\mathcal{A}$, we are interested in solving multi-robot problems using G. We assume here that the start and goal configurations of the simple robots are present as nodes in G (otherwise they can easily be added). The idea is that we seek paths in G along which the robots can go from their start configurations to their goal configurations, but we disallow simultaneous motions, and we also disallow motions along local paths that are blocked by the nodes at which the other robots are stationary. We refer to such paths as *G-discretised coordinated paths* (see also Figure 15). It can be shown that solving G-discretised problems (instead of continuous ones) is sufficient to guarantee *probabilistic completeness* of our multi-robot planning scheme, if the simple roadmaps are computed with *PPP* [46].

7.2 The super-graph approach

The question now is, given a simple roadmap $G = (V, E)$ for a robot $\mathcal{A}$, how to compute G-discretised coordinated paths for the composite robot $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ (with $\forall i : \mathcal{A}_i = \mathcal{A}$). For this we introduce the notion of *super-graphs*, that is,

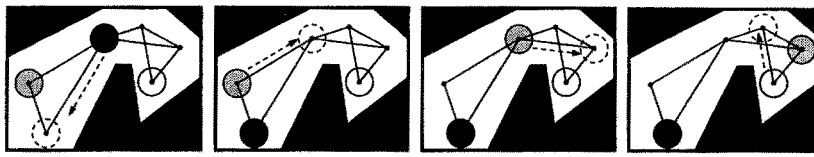


Fig. 15. A G-discretised coordinated path for 3 translating disc-robots.

roadmaps for the composite robots obtained by combining n simple roadmaps together. We discuss two types of such super-graphs. First, in Section 7.2, we describe a fairly straightforward data-structure, which we refer to as *flat super-graphs*. Its structure is simple, and its construction can be performed in a very time-efficient manner. However, its memory consumption increases dramatically as the number of robots goes up. For reducing this memory consumption (and, through this, increasing the planners power), we generalise this "flat" structure to a multi-level one, in Section 7.2. This results in what we refer to as *multi-level super-graphs*.

Using flat super-graphs In a *flat super-graph* $\mathcal{F}_G^n$, each node corresponds to a feasible placement of the n simple robots at nodes of G, and each edge corresponds to a motion of exactly one simple robot along a non-blocked local path of G. So $((x_1, \ldots, x_n), (y_1, \ldots, y_n))$, with all $x_i \in V$ and all $y_i \in V$, is an edge in $\mathcal{F}_G^n$ if and only if (1) $x_i \neq y_i$ for exactly one i and (2) (x_i, y_i) is an edge in E not blocked by any x_j with $j \neq i$. $\mathcal{F}_G^n$ can be regarded as the Cartesian product of n simple roadmaps. See Figure 16 for an example of a simple roadmap with a corresponding flat super-graph. Any path in the G-induced super-graph describes a G-discretised coordinated path (for the composite robot), and vice-versa. Hence, the problem of finding G-discretised coordinated paths for our composite robot reduces to graph searches in $\mathcal{F}_G^n$. A drawback of flat super-graphs is their size, which is exponential in n (the number of robots). For a formal definition of the flat super-graph method we refer to [46].

Using multi-level super-graphs The *multi-level super-graph method* aims at size reduction of the multi-robot data-structure, by combining multiple node-tuples into single super-nodes. While a node in a flat super-graph corresponds to a statement that each robot $\mathcal{A}_i$ is located at some particular *node* of G, a node in a multi-level super-graph corresponds to a statement that each robot $\mathcal{A}_i$ is located in some *subgraph* of G. But only subgraphs that do not *interfere* with each other are combined. We say that a subgraph A interferes with a subgraph B if a node of A blocks a local path in B, or vice versa. Due to space

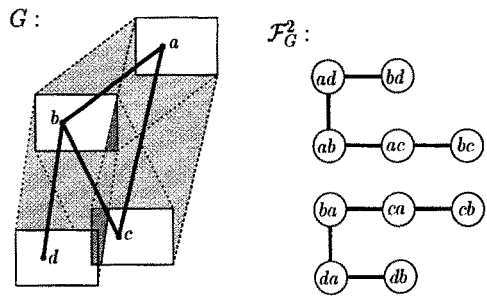


Fig. 16. At the left we see a simple roadmap G for the shown rectangular robot $\mathcal{A}$ (shown in white, placed at the graph nodes). We assume here that $\mathcal{A}$ is a translational robot, and the areas swept by the local paths corresponding to the edges of G are indicated in light grey. At the right, we see the flat super-graph $\mathcal{F}_G^2$, induced by G for 2 robots. It consists of two separate connected components.

limitations, we cannot go into much formal details regarding multi-level super-graphs. Here we will just describe the main points. The two main questions are how to obtain the subgraphs, and how to build a super-graph from these in a proper way.

For obtaining suitable subgraphs, we compute a recursive subdivision of the simple roadmap $G = (V, E)$, a so-called *G-subdivision tree* $\mathcal{T}$. Its nodes consist of connected subgraphs of G, induced by certain subsets of V. The root of $\mathcal{T}$ is the whole graph G. The children $(\tilde{V}_1, \tilde{E}_1), \ldots, (\tilde{V}_1, \tilde{E}_1)$ of each internal node $(\tilde{V}, \tilde{E})$ are chosen such that $\tilde{V} = \bigcup_{1 \le i \le k} \tilde{V}_i$ and $\bigcap_{1 \le i \le k} \tilde{V}_i = \emptyset$. Furthermore, all leafs, consisting of one node and no edges, lie at the same level of the tree $\mathcal{T}$. This of course in no way defines a unique G-subdivision tree. We just give a brief sketch of the algorithm that we use for their construction. After the root r (=G) has been created, a number of its nodes are selected heuristically, and subgraphs are grown around these "local roots", until all nodes of r lie in some subgraph. These subgraphs form the children of r, and the procedure is applied recursively to each of these. The recursion stops at subgraphs consisting of just one node, and care is taking to build a perfectly balanced tree.

For n robots, a simple roadmap $G = (V, E)$ together with a G-subdivision tree $\mathcal{T}$ uniquely defines a *multi-level super-graph* $\mathcal{M}_{G\mathcal{T}}^n$. A n-tuple $(X_1, \ldots, X_n)$ of equal-level nodes of $\mathcal{T}$ is a *node* of $\mathcal{M}_{G\mathcal{T}}^n$ if and only if all subgraphs X_i in the tuple are mutually non-interfering. We define the edges in $\mathcal{M}_{G,\mathcal{T}}^n$ in terms of the flat super-graph $\mathcal{F}_G^n$ induced by G. A pair of super-nodes $((X_1, \ldots, X_n), (Y_1, \ldots, Y_n))$ forms an *edge* E in $\mathcal{M}_{G,\mathcal{T}}^n$ if and only if there exists an edge $e = ((x_1, \ldots, x_n), (y_1, \ldots, y_n))$ in $\mathcal{F}_G^n$ with, for all $i \in \{1, \ldots, n\}$, x_i being a node of X_i and y_i being a node of Y_i. We refer to e as the *underlying*

flat edge of E. Also, for the $i \in \{1, \ldots, n\}$ with $x_i \neq y_i$, we refer to the simple robot $\mathcal{A}_i$ as the *active robot* of E (and to the others as the *passive* robots).

We want to stress here that the flat super-graph $\mathcal{F}_G^n$, which can be enormous for $n > 3$ (that is, more than 3 robots), is only used for definition purposes. For the actual construction of our multi-level graph $\mathcal{M}_{G\mathcal{T}}^n$ we fortunately need not to compute $\mathcal{F}_G^n$.

Simulation results show that the size of multi-level super-graphs is considerably smaller than that of equivalent flat super-graphs. Further size-reduction can be achieved by using what we refer to as *sieved* multi-level super-graphs. From experiments we have observed that the connectivity of the free configurations space of the composite robot is typically captured by only a quite small portion of $\mathcal{M}_{G\mathcal{T}}^n$, namely by that portion constructed from the relatively large subgraphs in $\mathcal{T}$. For this reason, we construct $\mathcal{M}_{G\mathcal{T}}^n$ incrementally. We sort the subgraphs in $\mathcal{T}$ by size, and pick them in reversed order of size. For each such picked subgraph we extend the super-graph $\mathcal{M}_{G,\mathcal{T}}^n$ accordingly. By keeping track of the connected components in $\mathcal{M}_{G\mathcal{T}}^n$ we can determine the moment at which the free space connectivity has been captured, and at this point the super-graph construction is stopped.

7.3 Retrieving the coordinated paths

Paths from multi-level super-graphs do not directly describe coordinated paths (as opposed to paths from flat super-graphs). For retrieving a coordinated path from a multi-level super-graph $\mathcal{M}_{G\mathcal{T}}^n$, first the start and goal configurations must be connected by coordinated paths to nodes X and Y of $\mathcal{M}_{G,\mathcal{T}}^n$. Such *retraction paths* can be computed by probabilistic motions of the simple robots. Then, a path $P_{\mathcal{M}}$, connecting X and Y in $\mathcal{M}_{G\mathcal{T}}^n$, must be found, and transformed to a coordinated path P. For each edge E in $P_{\mathcal{M}}$, we do the following: First, we identify the underlying simple edge $e = (a, b)$, and, within its subgraph, we move the active robot to a. Then, we move all passive robots to nodes within their subgraphs that do not block e. And finally we move the active robot to b (again within its subgraph), over the local path corresponding to e. Applied to all the consecutive edges of $P_{\mathcal{M}}$, this yields a coordinated path that, after concatenation with the two retraction paths, solves the given multi-robot path planning problem.

It follows rather easily from the definition of multi-level super-graphs that the described transformation is always possible.

7.4 Application to car-like robots

We have applied both the flat super-graph method as well as the multi-level super-graph method to *car-like* robots. We have implemented the planners in

C++, and tested them on a number of realistic problems, involving up to 5 car-like robots moving in the same environment. Below, we give simulation results from experiments performed with the multi-level super-graph method, for two different environments. The planner was again run on a Silicon Graphics Indigo^2 workstation with an R4400 processor running at 150 MHZ, rated with 96.5 SPECfp92 and 90.4 SPECint92 on the SPECMARKS benchmark.

For both scenes we have first constructed a simple roadmap with *PPP*. The sizes and densities of the two constructed simple roadmaps are sufficient to allow for the existence of G-discretised solutions to most non-pathetic problems in the scenes. Then, we have constructed the multi-level super-graphs incrementally by picking the subgraphs from the G-subdivision tree in order of decreasing size, as described in Section 7.2. We stopped the construction at the point were the multi-level super-graphs consisted of just one major component.

We report the sizes of the resulting super-graphs $\mathcal{M}^n_{GT} = (V_\mathcal{M}, E_\mathcal{M})$ and the time required for their construction. Also we give indications of the times required for retrieving and smoothing coordinated paths from the resulting super-graphs. Smoothing is quite essential for obtaining practical solutions, because the coordinated paths retrieved directly are typically very long and "ugly". We use heuristic algorithms for reducing the lengths of the coordinated paths (For details, see [46]).

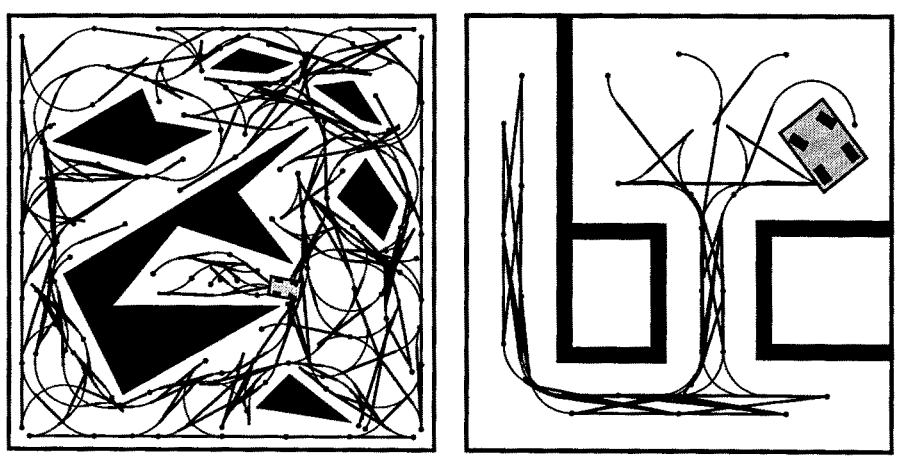


Fig. 17. Two scenes for the multi-robot path planner. Both scenes are shown together with a simple roadmap G for the indicated rectangular car-like robot. Not the edges, but the corresponding local paths are shown.

The left half of Figure 17 shows the first scene, together with the simple roadmap G, consisting of 132 nodes and 274 edges, constructed in about 14 seconds. In the table below we shown the sizes and the construction times of the induced multi-level super-graphs, for 3, 4, and 5 robots. Retrieving and smoothing coordinated paths required, roughly, something between 10 seconds (for 3 robots) and 20 seconds (for 5 robots). See Figure 18 for a path retrieved from the supergraph for 5 robots.

n	$\|V_{\mathcal{M}}\|$	$\|E_{\mathcal{M}}\|$	Time
3	408	2532	18.5
4	2256	15216	18.8
5	7080	33120	23.3

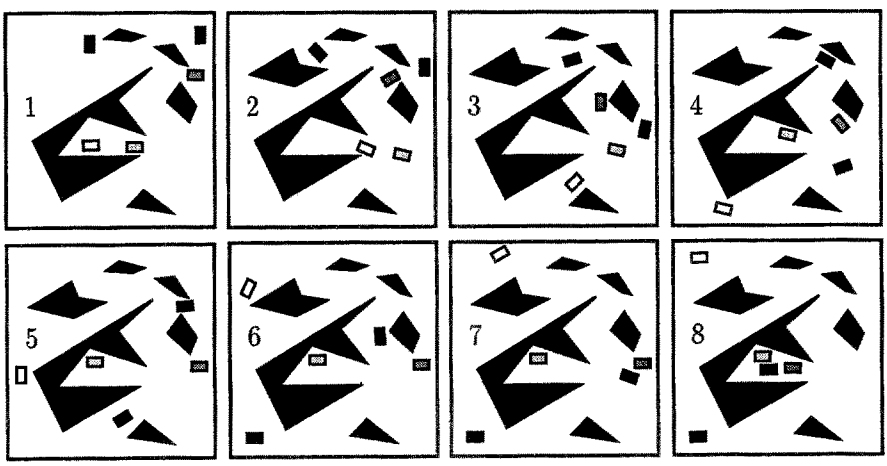


Fig. 18. Snapshots of a coordinated path in the first scene for 5 robots, retrieved from the multi-level super-graph.

The right half of Figure 17 shows the second scene on which we test the multi-robot planner. In the table below, the sizes and the construction times of the induced multi-level super-graphs, for 3 and 4 robots, are given. Here, retrieving and smoothing coordinated paths required was easier. Roughly, it took about 6 seconds for 3 robots and 8 seconds for 4 robots. See Figure 19 for a path retrieved from the supergraph for 4 robots.

n	$\|V_{\mathcal{M}}\|$	$\|E_{\mathcal{M}}\|$	Time
3	3018	15630	1.2
4	29712	152016	8.1

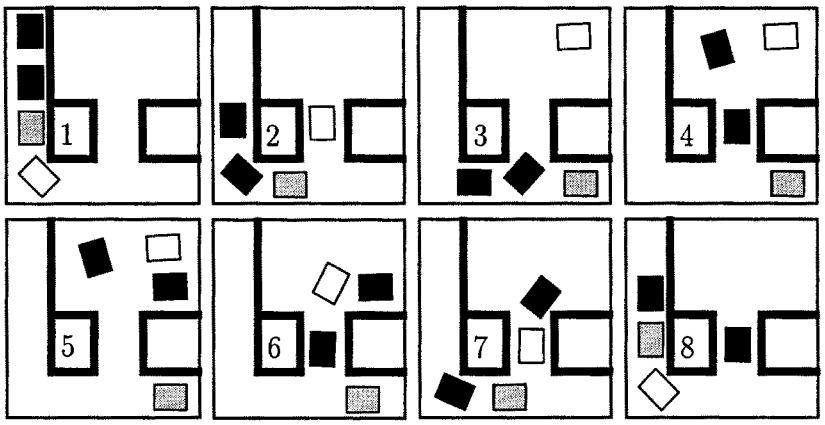


Fig. 19. Snapshots of a coordinated path in the second scene for 4 robots, retrieved from the multi-level super-graph.

We see that the data-structures in the second scene are considerably larger than those required for the first, although the first scene seems to be more complex. The cause for this must be that the compact structure of the free space in the second scene as well as the relatively large size of the robot cause more subgraphs to interfere. Hence, in the second scene, subdivision into smaller subgraphs is required.

7.5 Discussion of the super-graph approach

The presented multi-robot path planning approach seems to be quite flexible, as well as time and memory efficient. The power of the presented approach lies in the fact that only self-collision avoidance is dealt with for the composite robot, while all other (holonomic and nonholonomic) constraints are solved in the C-spaces of the simple robots.

There remain many possibilities for future improvements. For example, smarter ways of building the G-subdivision trees probably exist. For many applications, it even seems sensible to use characteristics of the workspace geometry for determining the subgraphs in the G-subdivision tree. Also, techniques for analysing the expected running times need to be developed.

We have seen that for up to 5 independent robots the method proves practical. However, in many applications one has to deal with much larger fleets of

mobile robots. Due to the enormous complexity of such systems, only decoupled planners can be used here. Decoupled planners however can fall into deadlocks. Centralised planners could be integrated into existing large scale decoupled planners for resolving deadlock situations in specific (local) workspace areas where these could arise. For example, if $\mathcal{R}$ is such an area, the global decoupled planner could enforce a simple rule stating that, at any time instant, no more than say 4 robots are allowed to be present in $\mathcal{R}$. Path planning within $\mathcal{R}$ can then be done by a centralised planner, like for example the planner presented in this section.

8 Conclusions

In this chapter an overview has been given on a general probabilistic scheme *PPP* for robot path planning. It consists of two phases. In the roadmap construction phase a probabilistic roadmap is incrementally constructed, and can subsequently, in the query phase, be used for solving individual path planning problems in the given robot environment. So, unlike other probabilistically complete methods, it is a learning approach. Experiments with applications of *PPP* to a wide variety of path planning problems show that the method is very powerful and fast. Another strong point of *PPP* is its flexibility. In order to apply it to some particular robot type, it suffices to define (and implement) a robot specific local planner and some (induced) metric. The performance of the resulting path planner can, if desired, be further improved by tailoring particular components of the algorithm to some specific robot type.

Important is that probabilistic completeness, for holonomic as well as non-holonomic robots, can be obtained by the use of local planners that respect certain general topological properties. Furthermore, there exist some recent results that, under certain geometric assumptions on the free C-space, link the expected running time and failure probability of the planner to the size of the roadmap and characteristics of paths solving the particular problem. For example, under one such assumption, it can been shown that the expected size of a probabilistic roadmap required for solving a problem grows only logarithmically in the complexity of the problem.

Numerous extensions of the approach are possible. One such extension has been described in this chapter, dealing with the multi-robot path planning problem. Other possibilities include, for example, path planning in partially unknown environments, path planning in dynamic environments (e.g., amidst moving obstacles), and path planning in the presence of movable obstacles.

References

1. J.M. Ahuactzin. *Le Fil d'Ariadne: Une Méthode de Planification Générale. Application á la Planification Automatique de Trajectoires.* PhD thesis, l'Institut National Polytechnique de Grenoble, Grenoble, France, September 1994.
2. R. Alami, F. Robert, F. Ingrand, and S. Suzuki. Multi-robot cooperation through incremental plan-merging. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 2573–2578, Nagoya, Japan, 1995.
3. J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. To appear in *Intern. Journal of Rob. Research.*
4. J. Barraquand and J.-C. Latombe. A Monte-Carlo algorithm for path planning with many degrees of freedom. In *Proc. IEEE Intern. Conf. on Robotics and Automation*, pages 1712–1717, Cincinnati, OH, USA, 1990.
5. J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Internat. Journal Robotics Research.*, 10(6):628–649, 1991.
6. J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.
7. P. Bessière, J.M. Ahuactzin, E.-G. Talbi, and E. Mazer. The Ariadne's clew algorithm: Global planning with local methods. In *Proc. The First Workshop on the Algorithmic Foundations of Robotics*, pages 39–47. A. K. Peters, Boston, MA, 1995.
8. S.J. Buckley. Fast motion planning for multiple moving robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 322–326, Scottsdale, Arizona, USA, 1989.
9. J.F. Canny. *The Complexity of Robot Path Planning.* MIT Press, Cambridge, USA, 1988.
10. M. Erdmann and T. Lozano-Pérez. On multiple moving objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1419–1424, San Francisco, CA, USA, 1986.
11. P. Ferbach. A method of progressive constraints for nonholonomic motion planning. Technical report, Electricité de France. SDM Dept., Chatou, France, September 1995.
12. C. Fernandes, L. Gurvits, and Z.X. Li. Optimal nonholonomic motion planning for a falling cat. In Z. Li and J.F. Canny, editors, *Nonholonomic Motion Planning*, Boston, USA, 1993. Kluwer Academic Publishers.
13. J. Hopcroft, J.T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the warehouseman's problem. *International Journal of Robotics Research*, 3(4):76–88, 1984.
14. Th. Horsch, F. Schwarz, and H. Tolle. Motion planning for many degrees of freedom - random reflections at C-space obstacles. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 3318–3323, San Diego, USA, 1994.
15. Y. Hwang and N. Ahuja. Gross motion planning—a survey. *ACM Comput. Surv.*, 24(3):219–291, 1992.

16. Y.K. Hwang and P.C. Chen. A heuristic and complete planner for the classical mover's problem. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 729–736, Nagoya, Japan, 1995.
17. B. Langlois J. Barraquand and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Trans. on Syst., Man., and Cybern.*, 22(2):224–241, 1992.
18. P. Jacobs, J.-P. Laumond, and M. Taïx. A complete iterative motion planner for a car-like robot. *Journees Geometrie Algorithmique*, 1990.
19. L. Kavraki. *Random networks in configuration space for fast path planning.* Ph.D. thesis, Department of Computer Science, Stanford University, Stanford, California, USA, January 1995.
20. L. Kavraki, M.N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. In *IEEE International Conference on Robotics and Automation*, pages 3020–3026, Minneapolis, MN, USA, 1996.
21. L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 2138–2145, San Diego, USA, 1994.
22. L. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Proc. 27th Annual ACM Symp. on Theory of Computing (STOC)*, pages 353–362, Las Vegas, NV, USA, 1995.
23. L. Kavraki, P. Švestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans. Robot. Autom.*, 12:566–580, 1996.
24. F. Lamiraux and J.-P. Laumond. On the expected complexity of random path planning. In *Proc. IEEE Intern. Conf. on Robotics and Automation*, pages 3014–3019, Mineapolis, USA, 1996.
25. J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, USA, 1991.
26. J.-P. Laumond, P.E. Jacobs, M. Taïx, and R.M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Trans. Robot. Autom.*, 10(5), October 1994.
27. J.-P. Laumond, S. Sekhavat, and M. Vaisset. Collision-free motion planning for a nonholonomic mobile robot with trailers. In *4th IFAC Symp. on Robot Control*, pages 171–177, Capri, Italy, September 1994.
28. J.-P. Laumond, M. Taïx, and P. Jacobs. A motion planner for car-like robots based on a mixed global/local approach. In *IEEE IROS*, July 1990.
29. Y.H. Liu, S. Kuroda, T. Naniwa, H. Noborio, and S. Arimoto. A practical algorithm for planning collision-free coordinated motion of multiple mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1427–1432, Scottsdale, Arizona, USA, 1989.
30. P.A. O'Donnell and T. Lozano-Pérez. Deadlock-free and collision-free coordination of two robotic manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 484–489, Scottsdale, Arizona, USA, 1989.
31. M.H. Overmars. A random approach to motion planning. Technical Report RUU-CS-92-32, Dept. Comput. Sci., Utrecht Univ., Utrecht, the Netherlands, October 1992.

32. M.H. Overmars and P. Švestka. A probabilistic learning approach to motion planning. In *Proc. The First Workshop on the Algorithmic Foundations of Robotics*, pages 19–37. A. K. Peters, Boston, MA, 1994.
33. P. Pignon. *Structuration de l'Espace pour une Planification Hiérarchisée des Trajectoires de Robots Mobiles.* Ph.D. thesis, LAAS-CNRS and Université Paul Sabatier de Toulouse, Toulouse, France, 1993. Report LAAS No. 93395 (in French).
34. J.A. Reeds and R.A. Shepp. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics*, 145(2):367–393, 1991.
35. J.H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proc. 25th IEEE Symp. on Foundations of Computer Science*, pages 144–154, 1985.
36. J.T. Schwartz and M. Sharir. Efficient motion planning algorithms in environments of bounded local complexity. Report 164, Dept. Comput. Sci., Courant Inst. Math. Sci., New York Univ., New York, NY, 1985.
37. J.T. Schwartz and M. Sharir. On the 'piano movers' problem: III. coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal obstacles. *International Journal of Robotics Research*, 2(3):46–75, 1983.
38. S. Sekhavat and J.-P. Laumond. Topological property of trajectories computed from sinusoidal inputs for nonholonomic chained form systems. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 3383–3388, April 1996.
39. S. Sekhavat, P. Švestka, J.-P. Laumond, and M.H. Overmars. Probabilistic path planning for tractor-trailer robots. Technical Report 96007, LAAS-CNRS, Toulouse, France, 1995.
40. S. Sekhavat, P. Švestka, J.-P. Laumond, and M.H. Overmars. Multi-level path planning for nonholonomic robots using semi-holonomic subsystems. To appear in *Intern. Journal of Rob Research.*
41. M. Sharir and S. Sifrony. Coordinated motion planning for two independent robots. In *Proceedings of the Fourth ACM Symposium on Computational Geometry*, 1988.
42. P. Souères and J.-P. Laumond. Shortest paths synthesis for a car-like robot. *IEEE Trans. Automatic Control*, 41:672–688, 1996.
43. H.J. Sussmann. Lie brackets, real analyticity and geometric control. In R.W. Brockett, R.S. Millman, and H.J. Sussmann, editors, *Differential Geometric Control Theory.* Birkhauser, 1983.
44. H.J. Sussmann. A general theorem on local controllability. *SIAM Journal on Control and Optimization*, 25(1):158–194, January 1987.
45. P. Švestka. A probabilistic approach to motion planning for car-like robots. Technical Report RUU-CS-93-18, Dept. Comput. Sci., Utrecht Univ., Utrecht, the Netherlands, April 1993.
46. P. Švestka and M.H. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 1631–1636, Nagoya, Japan, 1995.
47. P. Švestka and M.H. Overmars. Motion planning for car-like robots using a probabilistic learning approach. *Intern. Journal of Rob Research*, 16:119–143, 1995.

48. P. Švestka and M.H. Overmars. Multi-robot path planning with super-graphs. In *Proc. CESA'96 IMACS Multiconference*, Lille, France, July 1996.
49. P. Švestka and J. Vleugels. Exact motion planning for tractor-trailer robots. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 2445–2450, Nagoya, Japan, 1995.
50. D. Tilbury, R. Murray, and S. Sastry. Trajectory generation for the n-trailer problem using goursat normal form. In *Proc. IEEE Internat. Conf. on Decision and Control*, San Antonio, Texas, 1993.
51. P. Tournassoud. A strategy for obstacle avoidance and its application to multi-robot systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1224–1229, San Francisco, CA, USA, 1986.
52. S.M. La Valle and S.A. Hutchinson. Multiple-robot motion planning under independent objectives. To appear in *IEEE Trans. Robot. Autom.*.
53. F. van der Stappen. *Motion Planning amidst Fat Obstacles.* Ph.D. thesis, Dept. Comput. Sci., Utrecht Univ., Utrecht, the Netherlands, October 1994.
54. F. van der Stappen, D. Halperin, and M.H. Overmars. The complexity of the free space for a robot moving amidst fat obstacles. *Comput. Geom. Theory Appl.*, 3:353–373, 1993.

Collision Detection Algorithms for Motion Planning

P. Jiménez, F. Thomas and C. Torras

Institut de Robòtica i Informàtica Industrial, Barcelona

1 Introduction

Collision detection is a basic tool whose performance is of capital importance in order to achieve efficiency in many robotics and computer graphics applications, such as motion planning, obstacle avoidance, virtual prototyping, computer animation, physical-based modeling, dynamic simulation, and, in general, all those tasks involving the simulated motion of solids which cannot penetrate one another. In these applications, collision detection appears as a module or procedure which exchanges information with other parts of the system concerning motion, kinematic and dynamic behaviour, etc. It is a widespread opinion to consider collision detection as the main bottleneck in these kinds of applications.

In fact, static interference detection, collision detection and the generation of configuration-space obstacles can be viewed as instances of the same problem, where objects are tested for interference at a particular position, along a trajectory and throughout the whole workspace, respectively. The structure of this chapter reflects this fact.

Thus, the main guidelines in static interference detection are presented in Section 2. It is shown how hierarchical representations allow to focus on relevant regions where interference is most likely to occur, speeding up the whole interference test procedure. Some interference tests reduce to detecting intersections between simple enclosing shapes, such as spheres or boxes aligned with the coordinate axes. However, in some situations, this approximate approach does not suffice, and exact basic interference tests (for polyhedral environments) are required. The most widely used such test is that involving a segment (standing for an edge) and a polygon in 3D space (standing for a face of a polyhedron). In this context, it has recently been proved that interference detection between nonconvex polyhedra can be reduced, like many other problems in Computational Geometry, to checking some signs of vertex determinants, without computing new geometric entities.

Interference tests lie at the base of most collision detection algorithms, which are the subject of Section 3. These algorithms can be grouped into four approaches: multiple interference detection, swept volume interference, space-

time volume intersection, and trajectory parameterization. The multiple interference detection approach has been the most widely used under a variety of sampling strategies, reducing the collision detection problem to multiple calls to static interference tests. The efficiency of a basic interference test does not guarantee that a collision detection algorithm based on it is in turn efficient. The other key factor is the number of times that this test is applied. Therefore, it is important to restrict the application of the interference test to those instants and object parts at which a collision can truly occur. Several strategies have been developed: 1) to find a lower time bound for the first collision, 2) to reduce the pairs of primitives within objects susceptible of interfering, and 3) to cut down the number of object pairs to be considered for interference. These strategies rely on distance computation algorithms, orientation-based pruning criteria and space partitioning schemes.

Section 4 describes how motion planners adopt different strategies with respect to the use of static interference and collision detection procedures, depending on their global or local nature. While global planners use static interference tests, or their generalizations, to generate a detailed description of either configuration-space obstacles or free-space connectivity, incremental and local path planners avoid this costly computation by fully relying on collision detection tests during the search process.

Finally, some conclusions are sketched in Section 5.

2 Interference detection

Objects to be checked for interference are usually modeled by composing simple shapes. Hierarchies of spheres (or other primitive volumes) and polyhedral approximations are the most commonly used. The former exploit the low cost of detecting interference between spheres, which reduces to comparing the distance between their centers and the sum of their radii. This type of model is particularly adequate in situations not demanding high accuracy, since achieving that would require going down many levels in the hierarchy. Objects with planar faces and subject to small tolerances are usually dealt with using polyhedral representations of their boundaries.

Hierarchical approximations permit focusing on the regions susceptible of interfering, as described in Section 2.1. Then, basic interference tests, which are the subject of Section 2.2, need only be applied within the focused regions.

2.1 Focusing on relevant regions

The two main approaches to confine the search for interferences to particular portions of the solids are representation dependent. On the one hand, there are

algorithms that bound volume portions, and they are suited for volume representations, like Constructive Solid Geometry (CSG), octrees, or representations based on spheres. On the other hand, there are procedures that restrict the elements of the boundary of the objects that can intersect, and these algorithms are of course used together with boundary representations.

Hierarchical volume representations Two advantages of hierarchical representations must be highlighted:

- In many cases an interference or a non-interference situation can be easily detected at the first levels of the hierarchy. This leads to substantial savings under all interference detection schemes.
- The refinement of the representation is only necessary in the parts where collision may occur.

There are two types of bounding techniques for hierarchical volume representations, those that are based on an object partition hierarchy, and those where subregions of a space partition are considered.

Object partition hierarchies The so called "S-bounds" were developed and used in [8] for bounding spatially the part of the CSG tree that represents an intersection between two solids. S-bounds are simple enclosing volumes of the primitives at the leaves of the CSG tree: two examples are shown and discussed in [8] where rectangular parallelepipeds aligned with the coordinate axes and spheres are used as S-bounds. According to the set operations attached to every node in the tree, the S-bound corresponding to the root of the CSG intersection tree can be obtained after a number of pseudo-union and intersection operations of S-bounds. An algorithm that runs upwards and downwards on the tree performs all these operations (see Figs. 1 and 2). The main advantages of this procedure are the cut-off of subtrees included in empty bounds, leading to possibly important computational savings, and the focusing of intersection searching on zones where intersection can actually occur.

The "successive spherical approximation" described in [6] allows focusing on the region of possible interference by checking intersection of spherical sectors at different levels in the hierarchy (Fig. 3). Hierarchies based on spheres that bound objects at different levels of refinement are also used in [50] and in [52].

Space partition hierarchies The *octree* representation allows to avoid checking for collision in those parts of the workspace where octants are labelled *empty*, that is, where no part of any object exists. If a *full* (totally occupied by the solid) or *mixed* (partially occupied) octant is inside a *full* one of the other solid, interference occurs. Only if a *full* or *mixed*

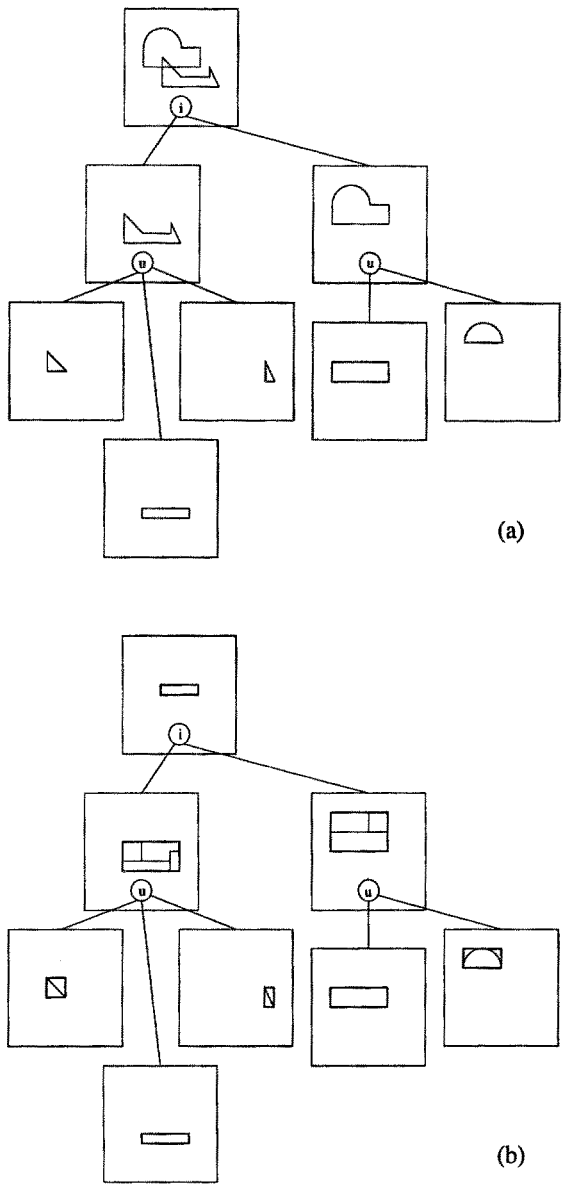


Fig. 1. *S-bounds.* (a) *The intersection (i) between two polygons described by their CSG representations has to be computed.* (b) *The rectangular boxes that bound the primitives are combined and the boxes corresponding to the higher levels are determined, according to the nature of the nodes (union or intersection).*

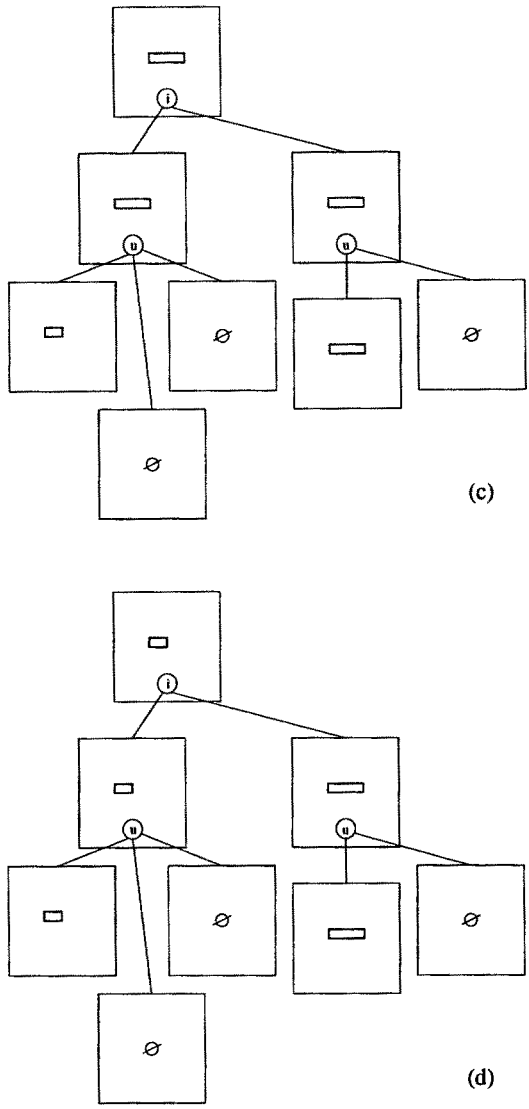


Fig. 2. *S-bounds (cont.).* (c) *The box obtained at the root node is intersected with the boxes of nodes at lower levels. The empty set is obtained for some nodes, which can be eliminated.* (d) *The representation is once again explored upwards, and a smaller box is obtained at the root. If the process is repeated once more every node will contain the small box or the empty set. This small box bounds the region where intersection has to be looked for.*

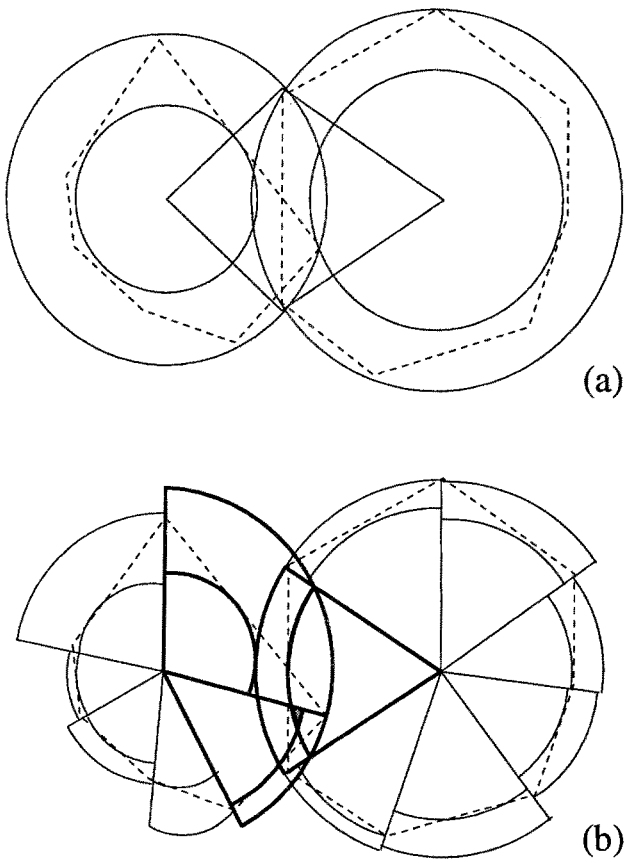


Fig. 3. (a) *Interference cannot be decided at the first level in the hierarchy, since neither the inner circles intersect nor the outer circles are disjoint. Nevertheless, the region of possible interference can be bounded, using the intersection points of the outer spheres.* (b) *At the next level two inner sectors intersect, thus interference exists.*

octant is inside a *mixed* one, the representation has to be further refined. The "natural" octree primitive is a cube [1,27], but there exist also models based on the same idea where spheres are used, as octant-including volumes [31] or within a different space subdivision technique, where the subdivision branching is 13 instead of 8 [39]. In the *binary space partition tree* [56], a binary tree is constructed that represents a recursive partitioning of space by hyperplanes. The authors describe such representation as a "crossing between octrees and boundary representations", but partitioning is not restricted to be axis-aligned, as in the octree representation, and therefore transformations (a change in orientation, for example) can be simply computed by applying the transformation to each hyperplane, without rebuilding the whole representation.

Boundary representations Hierarchical representations associated to bounding volumes that contain boundary features allow to restrict the effort of determining which parts of the objects boundaries may intersect to the most "promising" parts. Octrees have been used for subdividing axis aligned bounding boxes and constructing a bounding box hierarchy for the hull features (features of the polyhedron also appearing on its convex hull) and concavities of non-convex polyhedra [51]. Once penetration has been detected between the convex hulls of two polyhedra, a sweep and prune algorithm is applied to traverse the hierarchies down to the leaf level, where overlapping boxes indicate which faces may intersect, and exact contact points can be quickly determined.

In dense, cluttered environments, Oriented Bounding Boxes (OBB) perform better than axis aligned boxes or spheres, as they do fit more tightly to the objects and therefore less interferences between bounding volumes are reported. A hierarchical structure called OBB-Tree is used in [25] to represent polyhedra whose surfaces have been triangulated. Overlaps between OBBs are rapidly determined by performing 15 simple axis projection tests (about 200 arithmetic operations), as proved by the authors through their *separating axis theorem.*

2.2 Basic interference tests

Convexity plays a very important role in the performance of interference detection algorithms, and it is therefore used as classification criterion in the description below.

Convex polyhedra As pointed out in [37], intersection detection for two convex polyhedra can be done in linear time in the worst case. The proof is by reduction to linear programming, which is solvable in linear time for any fixed number of variables. If two point sets have disjoint convex hulls, then there is a

plane which separates the two sets. The three parameters that define the plane are considered as variables. Then, a linear inequality is attached to each vertex of one polyhedron, which specifies that the point is on one side of the plane, and the same is done for the other polyhedron (specifying now the location on the other side of the plane).

Moreover, convex polyhedra can be properly preprocessed, as described in [17], to make the complexity of intersection detection drop to $O(\log n \log m)$. Preprocessing takes $O(n+m)$ time to build a hierarchical representation of two polyhedra with n and m vertices. The lowest level in the hierarchical representation is a tetrahedron. At each level of the hierarchy, vertices of the original polyhedron are added, such that they form an independent set (i.e. , are not adjacent) in the polyhedron corresponding to this hierarchical level, and the corresponding edge and face adjacency relationships are updated.

In fact, this algorithm computes the distance between two convex polyhedra. Likewise, all algorithms developed for distance computation can be adapted to detect interference. We refer the reader to Section 3.2.

One convex and one one non-convex An algorithm for computing the intersection between a convex and a non-convex polyhedron is described in [45]. A by-product of intersection *computation* is interference *detection*. Let $\overline{P}$ and $\overline{Q}$ be the surface of P (convex, n edges) and of Q (possibly non convex, m edges), respectively. The algorithm needs to solve the *support problem*, that is, to determine at least one point of each connected component of $\overline{P} \cap \overline{Q}$ (this set of points will be called S). The methods for interference detection between convex polyhedra and linear subspaces developed by Dobkin and Kirkpatrick [16] are used for determining the intersections of $\overline{P}$ with edges and faces of Q: a hierarchical representation is used for P, so that the intersection between a line l, supporting an edge of Q, and $\overline{P}$ is computed in time $O(\log n)$, and a point in $h \cap \overline{P}$, where h is a plane supporting a face of Q, can also be computed in time $O(\log n)$. Therefore, an algorithm can be constructed that solves the support problem in $O(m \log n)$. The next step consists in determining $C = \overline{P} \cap \overline{Q}$, by taking points of S, which are intersections between a face and an edge, and determining the intersections between the face and the two faces which are adjacent to the edge. Finally, the segments of edges of P and Q which are *inside* the intersection have to be determined. Figure 4 illustrates the main steps of the strategy. The overall complexity is $O((n+m+s)\log(n+m+s))$, where s is the number of edges in the intersection.

Non-convex polyhedra: Decomposition into convex parts It is possible to extend the usage of the above algorithms to non-convex polyhedra just by decomposing these polyhedra into convex entities. Typically, decomposition

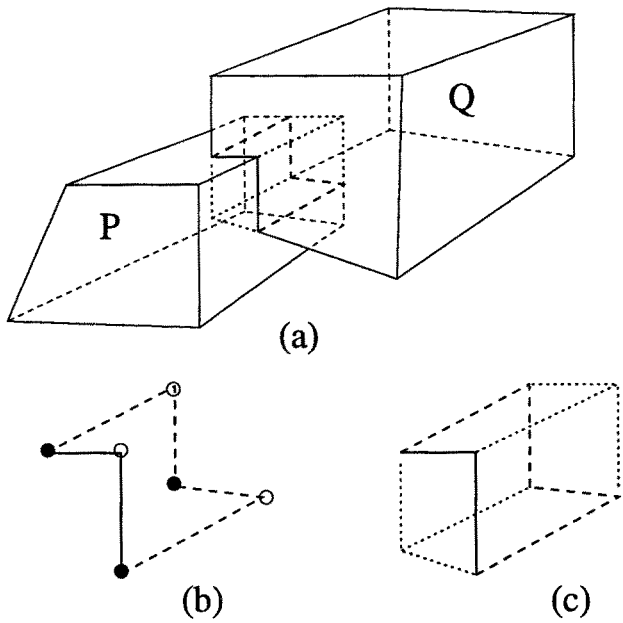


Fig. 4. (a) *Intersection computation (and -implicitly- interference detection) between two polyhedra, one of which is allowed to be non-convex (here, both have been depicted as convex for clarity).* (b) *Solving the support problem, the set of black (intersections of edges of Q and $\overline{P}$) and white (intersections of edges of P and $\overline{Q}$) points are obtained. Each pair of adjacent faces to these edges is intersected with the face of the other polyhedron that intersects this edge.* (c) *The segments of edges of one polyhedron inside the other one (and vice-versa) are finally computed (dotted lines).*

is performed in a preprocessing step, and therefore has to be computed only once. The performance of this step is a tradeoff between the complexity of its execution and the complexity of the resulting decomposition. For example, the extreme case of solving the *minimum decomposition* problem is known to be NP-hard in general [3]. On the other hand, algorithms such as that in [13] can always partition a polytope of n vertices into at most $O(n^2)$ convex entities.

Consider two polyhedra. Discarding the case in which one is fully inside the other, they intersect if their surfaces do. The detection of intersections between polyhedral surfaces reduces to detecting that an edge of one surface is piercing a face of the other surface.

Although interference detection becomes quite simple when faces are decomposed into convex polygons, and easy to implement, as explained below, the sequence of reductions used implies that the final complexity is $O(nm)$.

This reduction of the problem to detect edges piercing faces, formulated using the idea of *predicates* associated with the *basic contacts*, was introduced in [12]. The concept of basic contacts was introduced in [40], and its name derives from the fact that all other contacts can be expressed as a combination of them.

There are two basic contacts between two polyhedra. One takes place when a face of one polyhedron is in contact with a vertex of the other polyhedron (Type-A contact), and the other when an edge of one polyhedron is in contact with an edge of the other polyhedron (Type-B contact). Although in [40] and in [18] two different contacts between vertices and faces were considered, depending on whether they belong to the mobile polyhedron or the obstacle, avoiding to make this distinction greatly simplifies the presentation.

It is possible to associate a predicate with each basic contact, which will be true or false depending on the relative location between the geometric elements involved, as we will describe next.

Let us assume that face F_i is represented by its normal vector $\mathbf{f}_i$; edge E_j, by a vector $\mathbf{e}_j$ along it; and vertex V_k by its position vector $\mathbf{v}_k$. Although this representation is ambiguous, any choice leads to the same results in what follows.

According to Fig. 5(a), predicate $\mathbf{A}_{V_i,F_j}$, associated with a basic contact of Type-A, is defined as true when

$$\langle \mathbf{f}_j, \mathbf{v}_i - \mathbf{v}_k \rangle > 0, \tag{1}$$

for any vertex V_k in face F_j, and false otherwise.

According to Fig. 5(b), predicate $\mathbf{B}_{E_i,E_j}$, associated with a basic contact of Type-B, is defined as true when

$$\langle \mathbf{e}_i \times \mathbf{e}_j, \mathbf{v}_m - \mathbf{v}_k \rangle > 0, \tag{2}$$

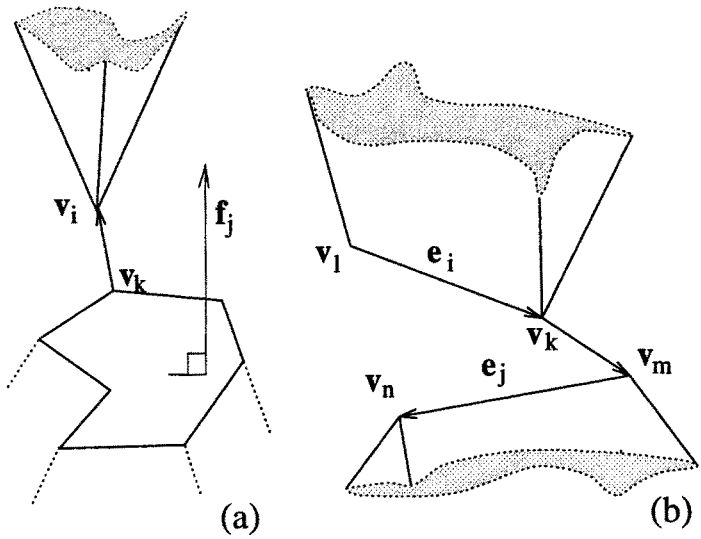


Fig. 5. *Geometric elements involved in the definition of the predicates associated with Type-A* (a) *and Type-B* (b) *basic contacts.*

V_m and V_k being one of the two endpoints of E_i and E_j, respectively, and false otherwise.

It can be checked that if one of the following predicates

$$\mathbf{O}^{out}_{E_i,F_j} = \neg\mathbf{A}_{V_x,F_j} \wedge \mathbf{A}_{V_y,F_j} \wedge \bigwedge_{E_k \in edges(F_j)} \mathbf{B}_{E_i,E_k}$$

$$\mathbf{O}^{in}_{E_i,F_j} = \mathbf{A}_{V_x,F_j} \wedge \neg\mathbf{A}_{V_y,F_j} \wedge \bigwedge_{E_i \in edges(F_j)} \neg\mathbf{B}_{E_i,E_k} \tag{3}$$

is true (see Fig. 6), then edge E_i intersects face F_j, provided that its edges (E_k) are traversed counter-clockwise.

Non-convex polyhedra: Direct approach If faces are not decomposed into convex polygons, two simple steps can be followed to detect whether an edge intersects a face. First, check if the edge endpoints are on opposite sides of the face plane. If so, check if the intersection point between the edge and the face plane is located inside the face by simply casting a ray from this point and determining how many times the ray intersects the polygon. Then, if this number is odd, the intersection does exists (odd-parity rule). Note that the

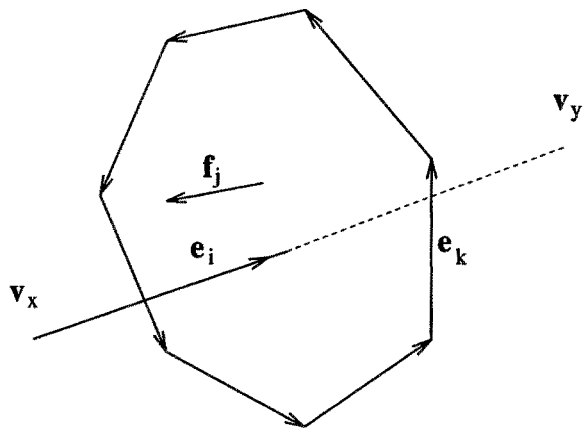


Fig. 6. *Basic edge-face intersection test (convex faces).*

latter step corresponds directly to solving a *point-in-polygon* problem, for which several alternatives, different from that of shooting a ray, have been proposed [28].

This was the approach adopted in [7] almost twenty years ago. Although the final complexity of the algorithm is clearly $O(nm)$, it is still the solution adopted in most implementations. Note that the only subquadratic algorithm developed so far [49] lacks practical interest because of the high time constants involved.

A simpler approach is to reduce the problem to computing the signs of some determinants [58], as it has been done for many other problems within the field of Computational Geometry [2],

Consider a face from one polyhedron, defined by the ordered sequence of the vertices around it, represented by their position vectors $\mathbf{p}_1, \ldots, \mathbf{p}_l$, expressed in homogeneous coordinates (that is, $\mathbf{p}_i = (p_{x_i}, p_{y_i}, p_{z_i}, 1)$), and an edge, from the other, defined by its endpoints $\mathbf{h}$ and $\mathbf{t}$. Then, consider a plane containing the edge and any other vertex, say $\mathbf{v}$, of the same polyhedron, so that all edges in the face whose endpoints are not on opposite sides of this plane are discarded. In other words, we define, according to Fig. 7, $s := \text{sign}\,|\mathbf{h}\ \mathbf{t}\ \mathbf{v}\ \mathbf{p}_i|$. Then, if $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ are on opposite sides, s should have a different sign from that of $|\mathbf{h}\ \mathbf{t}\ \mathbf{v}\ \mathbf{p}_{i+1}|$.

It can be checked that, if the number of edges straddling the plane and satisfying $s * sign\,|\mathbf{h}\ \mathbf{t}\ \mathbf{p}_i\ \mathbf{p}_{i+1}| > 0$ is odd, then the face is intersected by the edge. Actually, this is a reformulation of the odd parity rule that avoids the computation of any additional geometric entities such as those resulting from plane-edge or line-edge intersections.

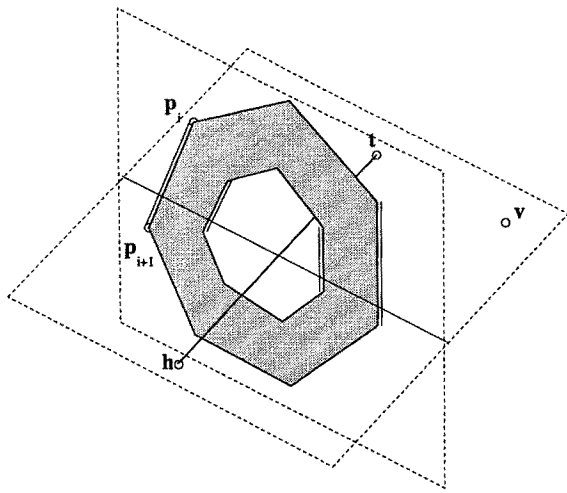


Fig. 7. *Basic edge-face intersection test (general faces).*

The two special cases in which the arbitrary plane intersects at one vertex of the face or it is coplanar with one of the edges lead to determinants that are null. Actually, equivalent situations also arise when the ray shooting strategy is used. In order to take them into account, a simple modification of the odd parity rule has to be introduced as in [7].

It is also interesting to point out that, if the arbitrary point $\mathbf{v}$ corresponds to a point on one of the two faces in which the edge lies, different from its endpoints, the above approach is a generalization of Canny's predicates, by simply noting that they can also be expressed in terms of signs of determinants involving vertex locations [58].

Thus, in order to decide whether two non-convex polyhedra intersect, only the signs of some determinants involving the vertex location coordinates are required. Since the signs of all the determinants involved are not independent, it is reasonable to look for a set of signs from which all other signs can be obtained. This is discussed in [57] through a formulation of the problem in terms of oriented matroids.

3 Collision detection

Collision detection admits several problem formulations, depending on the type of output sought and on the constraints imposed on the inputs. The simplest decisional problem, that looking for a yes/no answer, is usually stated as follows: Given a set of objects and a description of their motions over a certain

time span, determine whether any pair will come into contact. More intrincate versions require finding the time and features involved in the first collision, or even the time intervals over which objects would be intersecting if they were adhering to the predefined motions. Placing constraints on the inputs is a usual way of simplifying problems. Thus, often objects are assumed to be polyhedra, usually convex ones, and motions are constrained to be translational or quasi-linear.

The four main approaches that have been proposed to deal with the different instances of the collision detection problem are described in Section 3.1. After this description, it becomes clear that tests for static interference lie at the base of most approaches. However, the efficiency of a basic interference test does not guarantee that a collision detection algorithm based on it is in turn efficient. The other key factor is the number of times that this test is applied. Therefore, it is important to restrict the application of the interference test to those instants and object parts at which a collision can truly occur. Section 3.2 reviews the different strategies for time and space bounding that have been developed, among them distance computation, orientation-based pruning criteria, and prioritizing collision pairs.

3.1 Four main approaches

Collision detection algorithms can be grouped into four approaches: multiple interference detection, swept volume interference, extrusion in 4D space, and trajectory parameterization. As we will see, some approaches are linked to a particular object representation scheme (e.g. , extrusion is particularly suited to a CSG representation), while others do not.

Multiple interference detection The simplest way to tackle collision detection consists in sampling the trajectories followed by the objects and repeatedly applying a static interference test. This is called the multiple interference detection approach.

The way sampling is performed is crucial for the success of the approach. A too coarse sampling may lead to accepting a trajectory as safe when it actually leads to collision (see Fig. 8), while a too fine one may be computationally expensive. The reasonable way out is to apply *adaptive sampling.*

Ideally, the next time sample should be the earliest time at which a collision can really occur. The different sampling strategies differ in the way this earliest time is estimated. The most crude estimation is that relating a lower bound on the distance between objects to an upper bound on their relative velocities [10,15].

More sophisticated strategies take not only distance into account, but also directional information. One such strategy [23] requires computing the closest

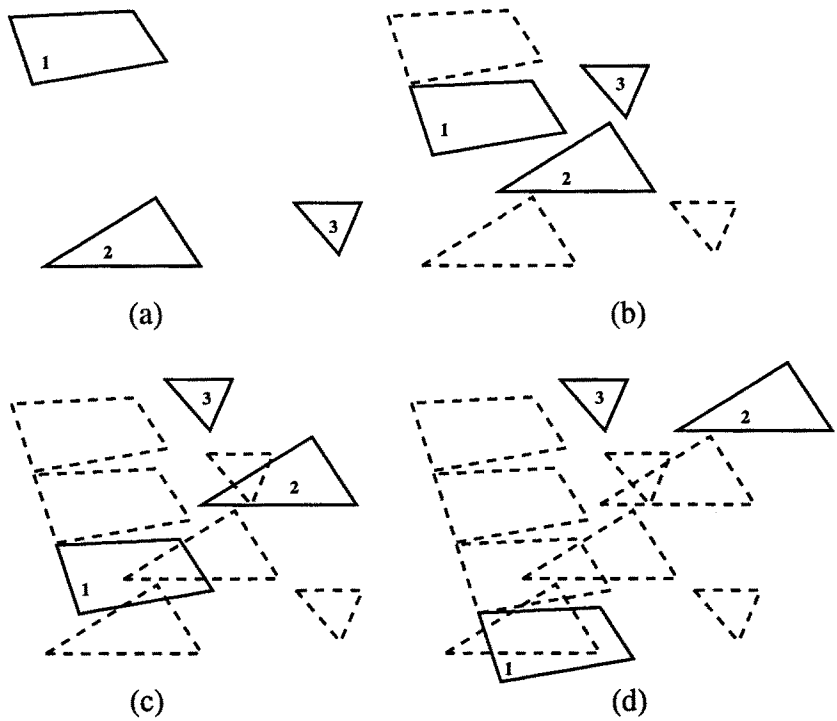


Fig. 8. *Multiple interference detection approach. As the time step is too large, the collision between the polygons 1 and 2, which takes place between instants* (b) *and* (c), *is missed. At instant* (d) *polygons 1 and 2 have attained their final positions, whereas polygon 3 had already attained it between instants* (b) *and* (c). *The polygons and their trajectories are the same as those in the next three figures.*

points from the objects at the current time sample, as well as the line joining them. The first future instant at which the projections of the objects on the line meet is taken as the next time sample (see Fig. 9).

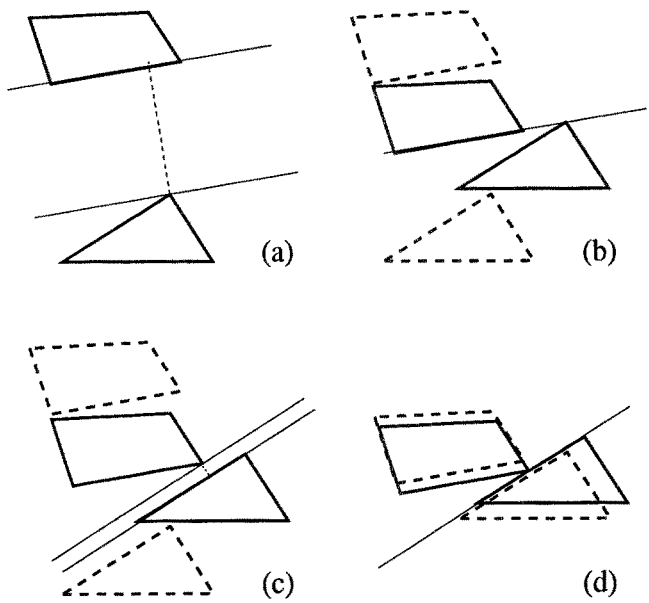


Fig. 9. *Adaptive time sampling. Starting position is depicted in* (a), *where the closest points and the line joining them are computed. The projections of the objects on this line meet at instant* (b), *which is taken as the next time sample. At this instant, the new closest points are computed* (c), *and the next time sample, where the polygons do actually collide, is determined in the same way* (d).

Since the closest points between two objects lie always in their boundaries, it is usual practice to resort to boundary representations (*B-rep*) when following a multiple interference detection approach. However, to confine the application of the interference test to those object parts susceptible of colliding first, spatial partitioning techniques such as octrees and voxels have also been used in conjunction with this approach.

Swept volume interference Given an object and a description of its motion over a time period, the volume containing all the points occupied by the object

at some time instant is called the *swept volume.* If the swept volumes for all the objects in a scene do not intersect, then no collision between them will occur during the specified time period. However, this is a sufficient, but not a necessary condition: It may happen that the swept volumes intersect but no collision takes place (see Fig. 10).

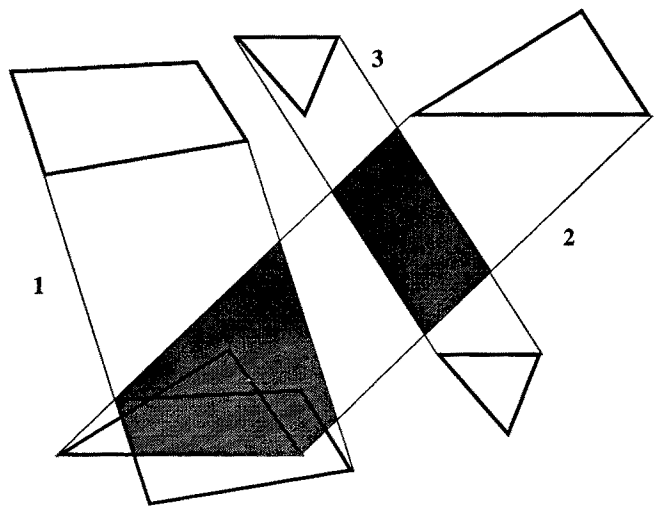


Fig. 10. *Swept volume interference. Polygons 1 and 2 collide, and their swept areas interfere. However, interference exists between the swept areas of polygons 2 and 3, but they do not actually collide.*

In order for the condition to be also necessary, the sweep has to be performed according to the *relative* motion of one object with respect to another one, for each pair of objects. This can be computationally very costly.

The generation of the swept volume per se is also computationally expensive. This is the reason why most works in this area deal with convex approximations of the swept volume and, only when the global swept volumes intersect, they proceed to split the trajectory into pieces and to compute a convex approximation of the swept volume for each piece [19].

The union of the convex approximations for the several trajectory pieces constitutes a much finer approximation to the real swept volume than the initial global approximation for the entire trajectory. For convex objects, Foisy and Hayward [19] have proved that the approximations obtained in the successive splittings of the trajectory converge to the real swept volume.

Simplifying alternatives consist in restricting the kind of shapes and trajectories to very simple ones [29], or creating implicitly the swept volume from the volumes swept out by the primitives of the *B-rep* [7].

Extrusion in 4D space Probably the collision detection approach most attractive from a theoretical viewpoint is that based on the extrusion operation [9]. Given an object and a description of its motion over a time period, the *extruded volume* is the spatio-temporal set of points representing the spatial occupancy of the object along its trajectory.

The intersection of two extruded volumes is a necessary and sufficient condition for the occurrence of a collision between the corresponding objects as they move along their respective trajectories (see Fig. 11). Therefore, this approach obviates a priori all the problems derived from sampling and from having to consider relative motions between pairs of objects. The problem that remains, however, is that of generating the volumes, which are 4D in this case.

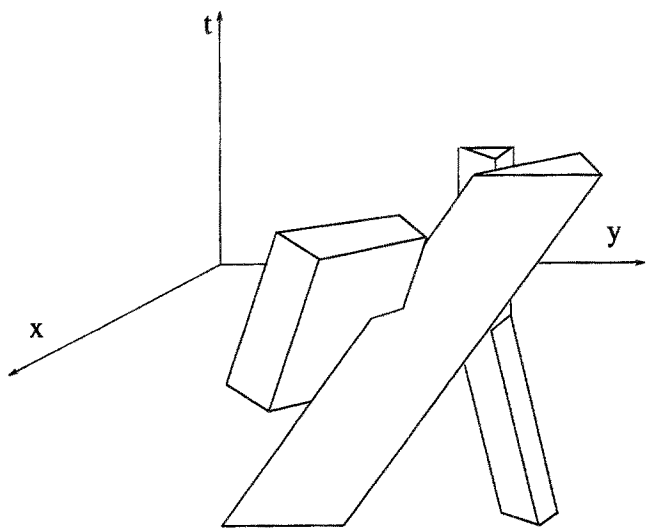


Fig. 11. *Interference between extruded volumes. Time is explicitly taken into account and therefore collision situations can be clearly identified (such as that of polygons 1 and 2). Note the change in the shape of the volume extruded by polygon 3, corresponding to the change in its velocity (it has stopped moving earlier than the other polygons).*

The extrusion operation is distributive with respect to the union, intersection and set difference operations. This motivated the development of the extrusion approach in the context of CSG representations. The mentioned distributive property guarantees that an object and its extruded volume can be represented through the same boolean combination of volumetric primitives and extrusions of these primitives, respectively.

The formal beauty of this approach is partially occluded by the high cost of its practical implementation. Thus, for example, the extrusion of a linear subspace subject to a constant angular velocity is bounded by a helicoidal hypersurface. For this reason, the implementation deals only with linear subspaces subject to piecewise translational motions [9].

Trajectory parameterization The collision instant can be analytically determined if the object trajectories are expressed as functions of a parameter (time) and the collision condition is formulated as a semialgebraic set involving the locations of object features (faces, edges and vertices). This requires to perform a change of variable in order to obtain an algebraic expression for rotation, instead of equations in terms of trascendent functions. By replacing those locations by the corresponding parameterized trajectories, a semialgebraic set in terms of a single variable (time) is obtained. Once this set is explicitly computed, the time instants at which objects establish and lose contact are known.

The trajectory parameterization approach has been followed in [12,33,54], where a polyhedra interference test is expressed as a combination of parameterized basic contact functions. These functions reflect the spatial relationships between the primitives of the *B-rep* of the polyhedra. The zeros of these functions delimit several time intervals, whose combination according to the interference test provides the desired set of intervals over which objects would be intersecting, if they were adhering to the predefined trajectories.

3.2 Strategies for space and time bounding

The first three approaches described in the preceding section eventually require to apply a static interference test between either 3D volumes or 4D ones. However, even if a basic interference test is made very efficient, the collision detection algorithm can still be computationally expensive if the basic test has to be applied many times. Thus, the key aspect of any collision detection scheme is to restrict as much as possible *when* and *where* this test is applied. Knowing how the objects are moving and how far away they are from one another, it is possible to bound the time interval where the collision is likely to occur. Therefore, it is important to determine quickly the distance between the objects. On the other hand, if the direction of motion is known, the search for

possible collisions can be restrained to those object parts which may first come into contact. Finally, if there are many moving objects in the scene, means to avoid having to check every pair of objects for collision need to be provided. These are the issues discussed in the next subsections.

Distance computation for collision time bounding Spherical representations are appealing because the elementary distance calculation between two spheres is trivial. The problem rather consists in determining which spheres of the representation have to be tested. In [59] the objects are described in terms of *spherical cones* (generated by translating a sphere along a line and changing its radius) and *spherical planes* (which are obtained by translating a sphere in two dimensions, and eventually changing also its radius). These primitives can also be viewed as a collection of spheres. Any distance can be expressed as a combination of the distances between two *spherical cones* and between a sphere and a *spherical plane*. The distance between two spherical cones is determined in two steps: first, compute the direction where the minimum distance occurs, then compute the involved spheres (locate their centers on the axes of the spherical cones). The distance between a sphere and a spherical plane is found by projecting the sphere perpendicularly on the plane, and calculating the sphere on the plane that corresponds to this projection. In any case, once the spheres are located, the distance is easily found as the distance between their centers minus the sum of their radius.

Most distance computation algorithms have been developed for convex polyhedra. Some exploit specific features of the polyhedra and therefore cannot be used for other type of geometric models. Others, like the method explained in [24], can be used with spherical [26] or other non-polytopal surface descriptions [22].

There are two main streams in the way that the distance computation problem is treated, namely the *geometric* and the *optimization* approaches.

The geometric approach

The closest points of two polyhedra are obtained, under this approach, by expanding a hierarchical (incremental) representation in a given direction or by navigating along the boundaries of the polyhedra. The euclidean distance between these closest points is then computed. The methods differ in the way that the closest points are obtained:

- An adequate representation may justify the effort spent in obtaining it, as a preprocessing step is done only once, if it allows important computational savings in subsequent operations. This is the idea behind Dobkin and

Kirkpatrick's hierarchical polyhedral representation, already mentioned in Section 2. Using their representation leads to distance computation in optimal worst-case $O(\log n \log m)$ time [17]. Every step of the closest points search procedure corresponds to a level in the construction of the hierarchical representation. In the first step the closest points of two tetrahedra (the lowest level in the hierarchical representation), have to be determined, which is trivial. Now consider the direction of the segment that joins the closest points found at a given step. The two planes which are perpendicular to this direction and touch each polyhedron (in the hierarchy expanded so far) bound the zone where the next closest pair has to be searched for. This zone consists, for each polyhedron, in the intersection of the next hierarchy level polyhedron and the negative halfspace defined by the plane (the normal of the plane points towards the polyhedron expanded so far). Thus, it is either a simplex or the empty set. If the closest points are not the same as in the previous step, then at least one of them belongs to one of these intersection simplices. Therefore, every search step is restricted to at most two simplices. The number of steps is bounded by $\log n \cdot \log m$. Figure 12 may help understand this procedure.

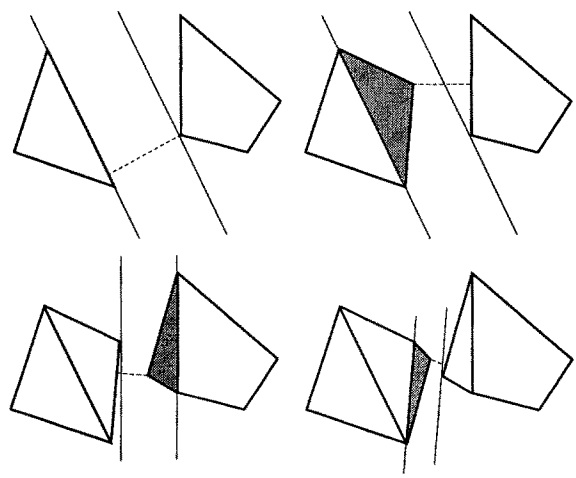


Fig. 12. *The hierarchical representation allows to build up and search only those parts of the polygons where the closest points can be found.*

- The Minkowski difference $M_{P,Q} = \{p - q | p \in P, q \in Q\}$ of two polytopes P and Q has been used in distance computation algorithms, since the distance between two polytopes is equal to the distance of their Minkowski difference to the origin (Fig. 13). This result is proved by Cameron and Culley (1986), and they provide also a procedure for computing $M_{P,Q}$, as well as the *minimum translational distance* (the minimum translation to be applied to one of the polyhedra in order to attain a situation where both polyhedra just touch). If $M_{P,Q}$ contains the origin of coordinates, the polyhedra are intersecting, and the minimum translational distance is negative.
- Efficiency is greatly increased in the procedure described in [24]. Complexity of $M_{P,Q}$ is, in general, quadratic, and therefore an algorithm that avoids generating the whole Minkowski difference would be desirable. Here, a directed sequence of subsets of the Minkowski difference polyhedron is generated, converging to a subset that contains the point that is closest to the origin. The convex hull of a subset of the vertices of the Minkowski difference is taken, and vertices are added which lie in the direction of interest, closer to the origin. At the same time non-relevant vertices are deleted, so that the search of the closest point to the origin is always done on a simplex, as can be seen in Fig. 14. The "vertex-selection" part of the algorithm can be done in linear time: a single direction is tested over the set of vertices of one of the original polyhedra and the opposite direction over the vertices of the other one.
- If a given point of a polyhedron is the closest one to a given feature (a vertex, an edge, or a face) of another polyhedron, it must be contained in the *Voronoi region* of this feature. The first step in this direction was done in [46] for rectangular boxes, but it was formalized and extended to any convex polyhedra in [37]. In their *incremental distance algorithm*, two arbitrary features are selected and the closest points that belong to them are obtained. In order to be actually the closest points of both polyhedra, these points have to belong to the Voronoi region of the other feature. If not, each point has to be closer to another neighboring feature, which is selected, and these steps are repeated until the condition of point-in-Voronoi-region-inclusion is met. The Voronoi regions for the three kinds of features are characterized in the mentioned reference (see Fig. 15).
  In their work, another important point is addressed: consider that the distance between two polyhedra has to be computed as they move along a finely discretized path. The closest features do not change often, and a change almost always involves neighboring features, due to the convexity of the polyhedra and the small discretization step. Therefore, not an arbitrary pair of features, but the closest features at the previous step are considered for initialization for every step. Simple preprocessing of the polyhedra, so that every feature has a constant number of neighboring features,

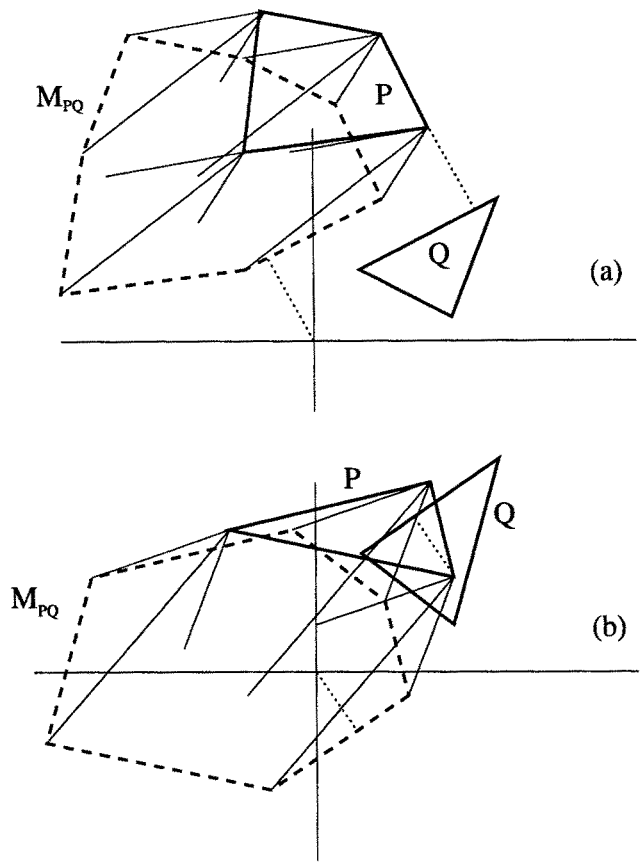


Fig. 13. (a) *The distance between the polygons is the same as the distance from the origin to their Minkowski difference.* (b) *If the polygons are interfering, the origin will be contained in the interior of their Minkowski difference. A hint is given for the construction of the Minkowski difference as the convex hulls of the points resulting from the subtraction of the vertices of Q from the vertices of P (thin lines).*

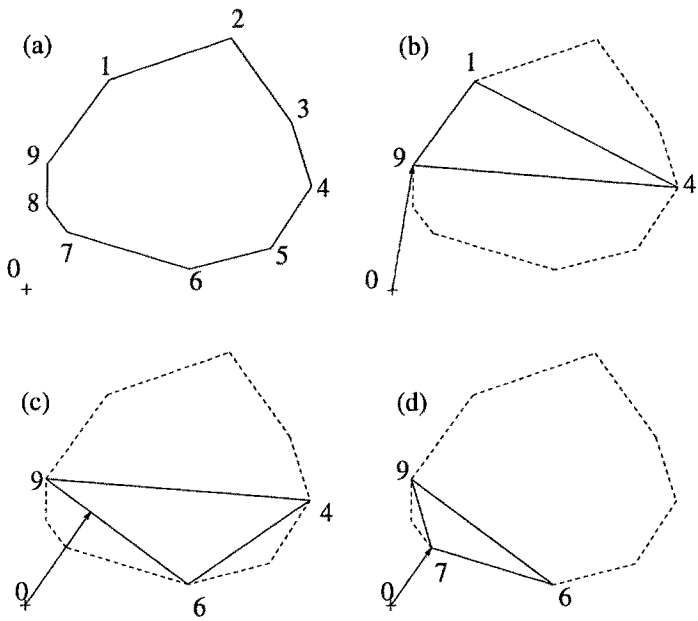


Fig. 14. *The closest point of the original Minkowski polygon* (a) *to the origin (0) has to be determined. The first simplex* (b) *has been chosen arbitrarily. A subset of vertices, whose convex hull contains the closest point of the simplex to the origin, is taken (4 and 9, although in this case also 1 and 9 could have been chosen), and a new vertex is selected* (c). *Note that this new vertex, 6, has the minimum projection onto the direction to the closest point found in* (b). *At the same time, the non-relevant vertex 1 is deleted. The direction to the closest point of the simplex in* (c) *is computed, and in the next step the closest point of the polygon to the origin (7 in this case) is determined* (d).

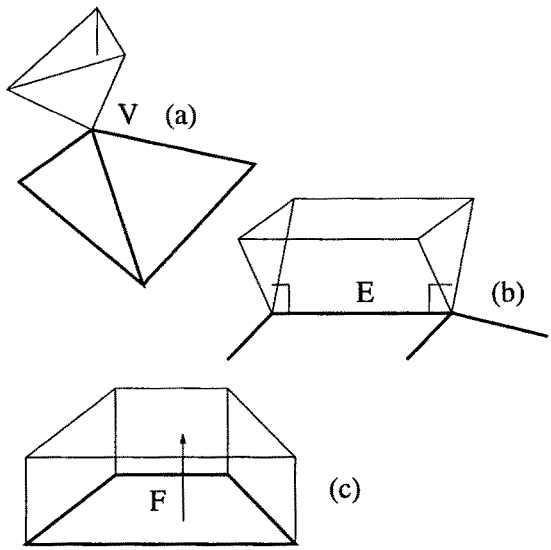


Fig. 15. *Voronoi regions of a vertex* (a), *an edge* (b), *and a face* (c).

allows the distance computation algorithm to run in expected constant time, once initialized (the global initialization step is typically linear in the total number of features).

To overcome the difficulty associated to the basic assumption that the two polyhedra have to be separated (if the objects actually penetrate each other, the algorithm goes into a cyclic loop), some authors have extended the space partition to the interior of the object, defining *pseudo*-Voronoi regions whose boundaries are faces determined by the centroid of the object and its edges (or the edges of its convex hull) [14,51,38]. These pseudo-Voronoi regions are only used to determine if the objects interpenetrate or not.

The optimization approach

Distance is viewed as a quadratic function to be minimized, under linear constraints due to the convexity of the polyhedra.

- The minimization of the non-linear function $f(p,q) = \|p-q\|^2/2$ subject to the linear constraints $\langle p, n_i^P \rangle \leq d_i^P, i = 1, \cdots, k^P$ and $\langle q, n_j^Q \rangle \leq d_j^Q, j = 1, \cdots, k^Q$ (these constraints mean that $p \in P$ and $q \in Q$, where the polyhedra P and Q are described as intersections of halfspaces) is solved in [5] by means of a gradient projection algorithm. At each step, the active

constraints are determined (those where equality holds, with certain tolerance) and Kuhn-Tucker conditions are used to test if the global minimum has been attained. If this is not the case, the coefficients of the Kuhn-Tucker conditions are used to find the new search direction. There are two alternatives for obtaining the starting points: to perform a simplex minimization subalgorithm along the direction given by the centroids of the polyhedra, or by considering the intersection points of the polyhedra boundary and the segment that joins the centroids.

- In applying Rosen's gradient projection method as Bobrow did, a convergence problem may occur, as stated in [62]. This problem is called the *zig-zag phenomenon* and it appears when the Kuhn-Tucker conditions are satisfied alternatively at each polyhedron. This happens because a zero vector is given as search direction on the polyhedron where the Kuhn-Tucker conditions are satisfied. The solution provided by these authors to this problem consists in considering as search direction for this polyhedron the projection of the search direction of the other polyhedron on the active constraints of the first one, instead of the zero vector, as shown in Fig. 16.

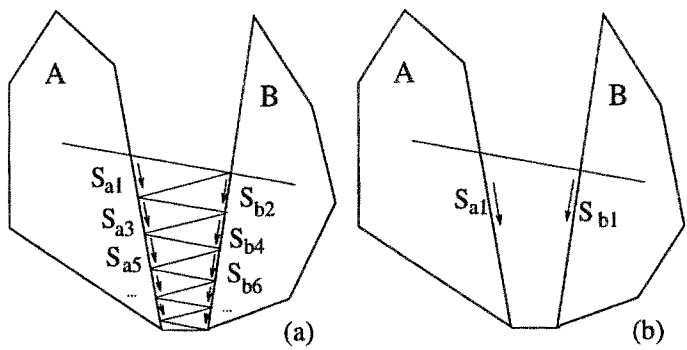


Fig. 16. *The zig-zag phenomenon* (a) *is avoided if the projection of* S_{a1} *on the active constraint of* B *is taken as the search direction* S_{b1} (b).

- Certain quadratic optimization problems can be solved in linear time, as shown in [44]. In [36] the distance computation problem between convex polyhedra is stated as described in Section 2, reducing it to a linear programming problem. It can be shown that the distance computation problem between non-intersecting convex polyhedra can be solved in $O(n)$ (n is the total number of vertices) by using a quadratic programming algorithm.

The linear constraints are here formulated in terms of the convex hull of the vertices of each polyhedron. In [53] the complexity associated with the intersection intensity computation between two polyhedra is also discussed.

No work has been devoted specifically to distance computation between non-convex polyhedra. In the context of collision detection, non-convex objects are usually approximated by simpler convex shapes, and a conservative lower bound on the distance is thus obtained. Some authors that deal with convex polyhedra mention the possibility of extending their algorithms to non-convex ones by decomposing them into convex entities, as explained in Section 2. Unfortunately, this solution may be inefficient, because of the complexity increment associated with the convex decomposition step. Moreover, if the number of generated convex entities is important, a large number of pairwise distances have to be computed, and although the individual objects are simpler, the net result is an important increment in the global complexity.

Orientation-based pruning If any kind of relative motion between two solids is allowed, every part of their boundary may intersect. But if a polyhedron can only move in a specific way with respect to another one, only certain parts of them can actually collide.

- Back-face culling techniques, which have been widely used in Computer Graphics to speed up the rendering of polyhedra, can also be used in the collision detection context to avoid unnecessary checking of boundary elements for collision, as shown in [61]. The basic idea consists in comparing the normal vectors of the faces of the polyhedra with the relative velocity vectors. A face is culled if its normal has a negative projection on the motion vector, as can be seen in Fig. 17. On the average, half of the faces of the two polyhedra are eliminated in this way. An algorithm that performs culling is described in the above reference.
- The incremental minimum distance realization technique [36] has already been mentioned in Section 3.2. At a given instant, the boundary elements that realize the minimum distance must be close to those realizing it at the previous instant, which are therefore taken as initial points for the search. In this case it is not a specific orientation, but a neighborhood criterion which is used for saving computational effort.
- A third possibility to avoid having to perform unnecesary intersection tests arises in the context of convex polyhedra where only translational motions are allowed. The *applicability constraints* [18] permit detecting those vertex-face and edge-edge pairings which can really come into contact (Fig. 18). The vertex-face applicability condition expresses the fact that a vertex can touch a face only if every adjacent edge projects positively on the

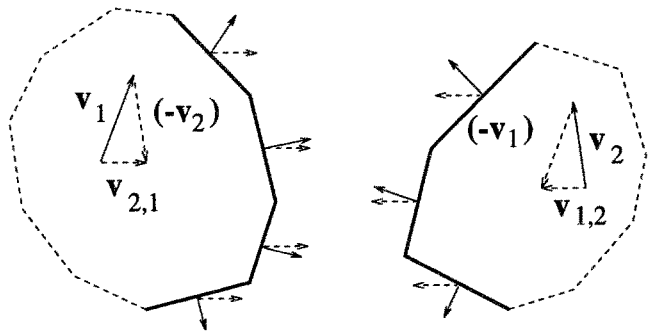


Fig. 17. *Only the faces (shown as heavy lines) whose normals have positive projections on the relative motion vectors ($v_{2,1}$ and $v_{1,2}$) need to be considered.*

face's normal (taking the vertex as origin of every edge interpreted as a vector). Two edges can touch only if there exist a separating plane between their respective *wedges*, as formally stated in the edge-edge applicability condition.

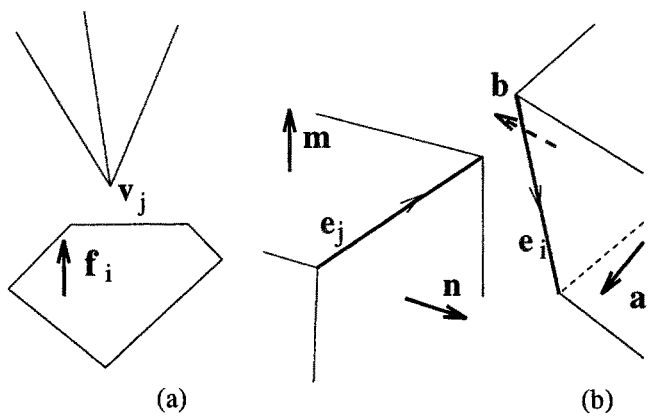


Fig. 18. (a) *An applicable vertex (V_j) - face (F_i) pairing.* (b) *Edges E_i and E_j are also applicable.*

The applicability constraints may be used as a preprocessing step in a collision detection scheme based on performing edge-face intersection tests. In general, if the contact between a vertex of a convex polyhedron and a face of

another polyhedron is applicable, only one of the edges which are adjacent to the vertex have to be considered for intersection with this face to report collision. Any other edge-face test with this face can be cut off. In a similar way one can bound the number of edges to be considered with respect to a given face resulting from edge-edge applicability constraints. In [32] an efficient algorithm for geometric pruning based on applicability constraints for convex polyhedra is described. Experimental results show that, by using this pruning technique, collision detection based on the edge-face intersection test has an expected $O(n)$ complexity, where n is the total number of edges, and the constant of linearity is close to 1. The algorithm is based on a face orientation graph representation, where face adjacency relations are explicitly depicted. The authors are currently working on extensions of the algorithm to non-convex polyhedra.

Prioritizing collision pairs The algorithms that try to avoid having to test for collision every possible pairing between solids in a given workspace are only useful if there is a large number of solids that may collide. Candidates for collision checking are prioritized in order to test only those pairs which are more likely to collide. The first criterion one may consider is distance, but it is not enough if the relative velocities are not taken into account, as pointed out in [20]. These authors introduce the concept of *awareness* or imminence of a collision. The shortest possible time at which a collision may occur is computed, considering mutual distance, instantaneous relative velocity, and velocity and acceleration bounds. This calculation is initially done for every pair, and afterwards the updating is done more frequently for those pairs whose awareness is larger. According to their value of awareness, the pairs are partitioned into equivalence classes whose collision imminence is similar. A binary partition scheme is used, where the cardinality of each class (called "bucket") is an increasing power of 2, and the value of the measure of awareness for all elements of a given class is greater than that for any other element in a lower bucket (of greater cardinality). At every time step, only one pair within each bucket is updated. Since the higher the bucket, the less pairs it contains, higher buckets are updated more frequently than lower ones. As their measures of awareness change, pairs can percolate from bucket to bucket.

In [47] a heap is used to store object pairs and soonest possible collision times, so that the pair on the top is the nearest to collide. This soonest collision time is computed from the distance between the closest points of the objects, current velocities and accelerations, and acceleration bounds assuming a ballistic trajectory for the objects. At each time step, integration of the dynamic state is done up to the time of collision for the pair on the top. Collision detection is performed for this pair, and if no collision actually occurs, the time

of impact is recomputed and the heap updated. Only the objects whose bounding boxes for their swept volumes during the frame period intersect with other boxes are selected and included in the heap. The intersections between these n boxes can be done in $O(n(1+\log R))$ (R is the ratio of largest to smallest box size), as shown in [48].

The same idea is followed in [35,14,51,38], where the concept of geometric and temporal coherence is emphasized, not only to speed up pairwise intersection detection (as done in [36] and whose algorithm is also used here) but also to perform less of these pairwise tests. If time steps (frames) are small enough, the position and orientation of the objects undergo only small changes, and it has already been mentioned how this fact can be used to keep track for the closest feature pair of two convex polyhedra. But it also means that there will be little changes in the position of the bounding boxes[1], and, of course, in the sequence of intervals that these bounding boxes project onto the coordinate axes, and which overlap (in the three axes) if and only if the corresponding bounding boxes intersect. Interval sorting techniques exist that take into account the sorted lists of interval endpoints in the previous frame, and allow to lower the effort to determine the projection intervals overlap to expected linear instead of $O(n \log n)$ (for n boxes). The computational cost of keeping track of changes in overlap status of interval pairs, following this line, is $O(n+s)$ (where s is the number of pairwise overlaps).

The so called *sweep algorithms* in [60] are along the same line: at a given instant, a plane is swept through the scene and only pairs of objects simultaneously intersecting the plane are tested for possible interference, thus avoiding to test every pair. The algorithm mentioned in this reference due to [30] does not find all intersections, although it reports at least one intersection if any exists, in $O(n \log^2 n)$ time between n spheres.

It is also possible to use such a sweep algorithm in 2D for bounding collision pairs in 3D, as done in [31]. 4D hyper-trapezoids are used to bound the object during its motion. If one intersection between two hyper-trapezoids occurs, the corresponding objects are tested for collision. These intersections are computed from intersections between their faces. The problem is reduced, by succesive projections, to a 2D segment intersection detection problem. The 2D sweep algorithm is described in [4] and runs in $O((m+k)\log m)$ time for m segments that intersect k times. Although for N objects the worst case value of k is $O(N^2)$, empirical evidence shows that the average value of k is much lower (0.07%).

[1] Two kinds of axis aligned bounding boxes are used in [14], fixed bounding cubes and dynamically rectangular boxes; for the latter, object orientation changes translate into changes in the dimensions of the bounding box.

4 Collision detection in motion planning

The goal of motion planning is to generate a collision-free path for a robot. Thus, collision-free trajectory planners must be able to perform some kind of geometric reasoning concerning collision detection between the robot and the obstacles [5,12]. In the generation of the path from the initial to the final robot configuration other criteria than mere collision avoidance may intervene, in order to optimize the resulting path in terms of its length, distances to obstacles, or orientation changes. Not to speak about extensions of the basic motion planning problem, that include uncertainty, kinematic constraints, or movable objects [34].

Depending on whether the static interference or collision detection tests are performed in a preprocessing step or during the path planning process, three kinds of planners can be distinguished: global, incremental, and local planners.

4.1 Global planners

In general, the configuration of a robot is given by a set of parameters, or degrees of freedom, that determine its location and orientation. The space defined by the ranges of allowed values for these parameters is usually called C-space.

An obstacle in C-space (C-obstacle, for short) is defined as the connected set of configurations where a given mobile object intersects with an obstacle in workspace. C-obstacles can be interpreted as the intersection of halfspaces bounded by C-surfaces, each C-surface being associated with a basic contact (see Section 2.2).

It can be shown that when working with polyhedra (and vertex, edge and face locations are expressed in terms of the degrees of freedom of the moving polyhedron), expressions (1) and (2) in Section 2.2 lead to the above-mentioned halfspaces, and using the predicate formalism in expression (3) a proper description of the C-obstacles can be obtained.

The collection of all C-obstacles constitutes the C-obstacle region. Some properties of the C-obstacles concerning compactness, connectedness and regularity are shown in [34]. C-obstacle generation can be viewed as a further generalization of the static interference and collision detection problems: here objects are not tested for interference at a particular configuration nor even along a given parameterized trajectory, but rather at all possible configurations in the workspace. Thus, once the C-obstacles are obtained, all information concerning interferences is captured.

Global planners construct a complete representation of the connectivity of free space (the complementary of the C-obstacle region) for their planning purposes. Several techniques have been devised to this end, depending on the

degrees of freedom of the robot as well as on its shape and the shape of the obstacles. Nevertheless, they are only of practical interest in low-dimensional configuration spaces. Pioneer work in this direction was done in [42,40] for polytopal environments. As a result of applying these techniques, a graph-based representation of free space is obtained: a roadmap or the connectivity graph of a cell decomposition. Afterwards a graph search algorithm can be applied in order to find the path that connects the initial and the final configurations.

In some simple cases, the configuration of a robot can be expressed in terms of the workspace coordinates of a given robot's point: for example, if the robot is a sphere (a disc in 2D) this reference point will be its center. The C-obstacles are trivially obtained from the obstacles in the workspace by performing an isotropic growth by the radius of the robot. Another simple case consists in a polytopal robot translating amidst polytopal obstacles. Any vertex of the robot can be taken as reference point. In this particular case, C-obstacles can also be interpreted as the Minkowski difference between the obstacle and the robot at a fixed orientation (as already mentioned in the context of distance computation in Section 3.2). This fact can be used for constructing the C-obstacle itself. This alternative representation is obviously related to the general predicate-based one, in the sense that the differences between the vertices corresponding to the features related to basic contacts are vertices of the C-obstacle. Both representations have been developed for convex polytopes. Non-convex obstacles can be treated in the same way by representing them as overlapping convex parts.

When the robot polytope is allowed to rotate, the computation of the C-obstacles becomes much more difficult. Although an approximate solution can be readily obtained by sampling the involved rotations, in general C-obstacles can only be accurately described using the aforementioned predicates, which can be formally interpreted as semialgebraic sets (see [11] for more details). Note that when all degrees of freedom but one are sampled, the problem becomes one of detecting intervals of interference, as many times as needed, depending on the sampling rate. This technique is used with up to three degrees of freedom in [41].

4.2 Incremental planners

While global path planners generate a detailed description of the connectivity of the whole free space, incremental path planners avoid this costly computation by obtaining this description in an incremental fashion. In this case, the construction of the free-space representation is carried out simultaneously with the path planning process. A paradigmatic example of this strategy can be found in [21], where a restricted visibility graph in C-space is built up iteratively. This subset of the whole visibility graph is granted to contain the optimal path. It is constructed by determining which C-obstacles intersect the

segments of the shortest path found so far (a straight line joining the initial and final positions at the first step), and rearranging the visibility graph with these new C-obstacles.

Randomized path planning methods might work in a similar way: points are randomly generated and those lying in free space are retained. Then, attempts are made to link these points by means of collision-free line segments. In this way, a representation of free space is gradually built up by locally testing for collisions, while generating a path from the initial to the final configurations. The same applies to those techniques that combine a potential field approach with randomization to escape from local minima. More details on this kind of algorithms can be found in Chapter 5.

4.3 Local planners

Local planners use collision detection as a subroutine whose output is used on-line to guide the search of a collision-free trajectory. The main difference with respect to incremental strategies is that local methods perform path planning by applying motion operators that act locally. In [18] these operators are used for sliding on C-surfaces and along their intersections without computing an explicit representation of free-space. They also allow to jump in free-space between obstacles. In any case, each time a C-surface is traversed, an interference test is performed to ensure that the motion is collision-free. These operators are the building blocks of more sophisticated *local experts*, which are strategies for deciding which trajectory to follow, based on the local geometry as well as on the history of the current planning process. This combination of motion in free space with motion in contact (or up to a safety distance from the obstacles) is used by other local motion planners. This is the case of the algorithms developed for planar articulated and 3D cartesian robot arms in [43,55]. The intersection points of the obstacles with the *main line* joining the initial and final positions are found, and motions along the obstacle boundaries between these points are computed.

Some local planners, as well as some incremental ones, can be applied in a recursive way: starting from an initial guess for a path between the initial and the final position, intermediate points are determined as collisions are detected, and the algorithm is recursively applied to the resulting path segments until a collision-free trajectory is detected or the conclusion is drawn that no such path exists.

While global path planners are always complete, that is, they are able to find a solution if one exists, those based on local techniques only ensure completeness at a resolution level. In [18] a partition of C-space based on *neighborhoods* is adopted, which are marked as visited if they are traversed by a trajectory

generated during the path planning process. As a consequence, if neighborhoods are made arbitrarily small, the algorithm becomes arbitrarily slow.

5 Conclusions

The different approaches to collision detection lie within two main categories: algebraic and geometric. The first try to solve equations that describe collision situations. These equations are expressed in terms of one parameter which is time or a variable related to time, and collision instants are determined. The trajectory parameterization approach corresponds to this strategy.

The geometric approaches compute geometric entities where time is treated as one more dimension, and try to determine intersections between them using methods developed within Computational Geometry. The most general approach is spatio-temporal volume intersection. However, no techniques exist for solving this problem directly, except for simple particular cases. The other two approaches can be viewed as particular techniques that simplify the resolution of the problem: the multiple interference detection approach applies sampling, whereas the swept volume interference approach uses projection. The drawbacks of these simplifying techniques have already been mentioned: sampling is complete only up to a given resolution, and projections may lead to report false collisions. Combinations of these techniques may allow to avoid these drawbacks, as in the adaptive sampling approach.

This perspective permits to formulate extensions for further work in a straightforward way: simplifying techniques have always been formulated considering time as a privileged dimension. Time is discretized by sampling or obviated by projection, but both techniques may also apply as well to the other dimensions or to combinations of them. To identify the classes of situations where sampling or projecting along other dimensions will ease the computation of collisions is more than an interesting theoretical exercise and may open new promising trends in collision detection algorithms.

Algebraic methods can also be viewed as a simplification of the general spatio-temporal problem formulation, as a projection on the time coordinate axis. Other dimensions instead of time could be used as parameters of the contact equations. However, the degree of the equations cannot be lowered in this way, and thus the efforts in looking for more efficient algorithms have to point in another direction, namely reducing the number of equations to be considered. This can be done by applying the complexity reduction techniques already mentioned in Section 3.2. In particular, orientation-based pruning may be applied to subdivide the trajectory into intervals where the same boundary primitives have to be considered for possible intersection, reducing drastically the number of contact equations to be considered within each interval.

In the line of developing complexity reduction techniques for interference detection we have centered our contribution to the PROMotion Project. Little work had previously been done on algorithms that deal directly with non-convex polyhedra, without decomposing them into convex parts. We have developed one such algorithm, based on a boolean combination of signs of vertex determinants [58]. Thus, neither line-plane intersections, nor ficticious edges and faces arising from a decomposition are required. Only the signs of determinants, for which there exist very efficient and robust algorithms, need to be computed. Moreover, we have developed a representation that captures the applicability relationships between the boundary features of two general polyhedra, that is, it allows to determine quickly which contacts can arise under translational motions [32]. In the case of non-convex polyhedra a large subset of all contacts that cannot take place for a given relative orientation are pruned off (all of them in the case of convex polyhedra). As these relationships hold over whole ranges of orientations, this technique can also be used to perform pruning along a trajectory that includes rotation[33], as mentioned above in the context of algebraic techniques for collision detection.

The speed-up of the basic interference and collision detection tests will necessarily improve the performance of motion planners, thus making the famous bottleneck a little bit wider.

References

1. N. Ahuja, R. T. Chien, R. Yen and N. Bridwell, "Interference detection and collision avoidance among three dimensional objects" in *I Annual National Conference on AI* (Stanford University) pp. 44–48 (Aug. , 1980).
2. F. Avnaim, "Evaluating signs of determinants using single-precision arithmetic" (Tech. Rep. INRIA 2306) (1994).
3. C. Bajaj and T. Dey, "Convex decomposition of polyhedra and robustness" in *SIAM J. Comput.* 21(2) pp. 339–364 (Apr. , 1992).
4. J. L. Bentley and T. A. Ottmann, "Algorithms for reporting and counting geometric intersections" in *IEEE Trans. Comput.* 28 (9) pp. 643–647 (Sept. , 1979).
5. J. E. Bobrow, "A direct optimization approach for obtaining the distance between convex polyhedra" in *Internat. J. Robotics Res.* 8 (3) pp. 65–76 (June, 1983).
6. S. Bonner and R. B. Kelley, "A representation scheme for rapid 3-D collision detection" in *Proceedings IEEE International Symposium on Intelligent Control* (Arlington (VA)) pp. 320–325 (Aug. , 1988).
7. J. W. Boyse, "Interference detection among solids and surfaces" in *Comm. ACM* 22 (1) pp. 3–9 (Jan. , 1979).
8. S. A. Cameron, "Efficient intersection tests for objects defined constructively" in *Internat. J. Robotics Res.* 8 pp. 3–25 (Feb. , 1989).
9. S. A. Cameron, "Collision detection by four-dimensional intersection testing" in *IEEE Trans. Robotics Automat.* 6 (3) pp. 291–302 (June, 1990).
10. S. A. Cameron, "A study of the clash detection problem in robotics" in *IEEE Proc. Int. Conf. Robotics Automat.* 1 (Saint Louis (MO)) pp. 488–493 (Mar. , 1985).
11. J. F. Canny, "The complexity of robot motion planning" (PhD Thesis, The MIT Press, Cambridge (MA)) (1988).
12. J. F. Canny, "Collision detection for moving polyhedra" in *IEEE Trans. Patt. Anal. Mach. Intell.* 8 (2) pp. 200–209 (Mar. , 1986).
13. B. Chazelle, "Convex partitions of polyhedra: A lower bound and a worst-case optimal algorithm" in *SIAM J. Comput.* 13 pp. 488–507 (1984).
14. J. D. Cohen, M. C. Lin, D. Manocha and M. K. Ponamgi, "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments" in *Proceedings of ACM Int. 3D Graphics Conference* 1 pp. 189–196 (1995).
15. R. K. Culley and K. G. Kempf, "A collision detection algorithm based on velocity and distance bounds" in *IEEE Proc. Int. Conf. Robotics Automat.* 2 (San Francisco (CA)) pp. 1064–1069 (Apr. , 1986).
16. D. Dobkin and D. Kirkpatrick, "Fast detection of polyhedral intersections" in *Lect. Notes in Comp. Sci.* (Springer-Verlag, New York–Heidelberg–Berlin) (140) pp. 154–165 (1982).
17. D. Dobkin and D. Kirkpatrick, "Determining the separation of preprocessed polyhedra -a unified approach" in *ICALP-90, Lect. Notes in Comp. Sci.* (Springer-Verlag, New York–Heidelberg–Berlin) (443) pp. 400–413 (1990).
18. B. R. Donald, "Local and global techniques for motion planning" (Masters Thesis, Massachusetts Institute of Technology) (1984).

19. A. Foisy and V. Hayward, "A safe swept volume method for collision detection" in *The Sixth International Symposium of Robotics Research* (Pittsburgh (PE)) pp. 61–68 (Oct. , 1993).
20. A. Foisy, V. Hayward and S. Aubry, "The use of awareness in collision prediction" in *IEEE Proc. Int. Conf. Robotics Automat.* 1 (Cincinnati (OH)) pp. 338–343 (May, 1990).
21. L-C. Fu and D-Y. Liu, "An efficient algorithm for finding a collision-free path among polyhedral obstacles" in *J. Robotic Sys.* 7 (1) pp. 129–137 (Feb. , 1990).
22. E. G. Gilbert and C-P. Foo, "Computing the distance between general convex objects in three-dimensional space" in *IEEE Trans. Robotics Automat.* 6 (1) pp. 53–61 (Feb. , 1990).
23. E. G. Gilbert and S. M. Hong, "A new algorithm for detecting the collision of moving objects" in *IEEE Proc. Int. Conf. Robotics Automat.* 1 (Scottsdale (AR)) pp. 8–14 (May, 1989).
24. E. G. Gilbert, D. W. Johnson and S. Keerthi, "A fast procedure for computing the distance between complex objects in three dimensional space" in *IEEE J. Robotics Automat.* 4 (2) pp. 193–203 (Apr. , 1988).
25. S. Gottschalk, M. C. Lin and D. Manocha, "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection" in *Proc. of ACM Siggraph'96* (1996).
26. G. J. Hamlin, R. B. Kelley and J. Tornero, "Efficient distance calculation using the spherically-extended polytope (S-tope) model" in *IEEE Proc. Int. Conf. Robotics Automat.* 3 (Nice (France)) pp. 2502–2507 (May, 1992).
27. V. Hayward, "Fast collision detection scheme by recursive decomposition of a manipulator workspace" in *IEEE Proc. Int. Conf. Robotics Automat.* 2 (San Francisco (CA)) pp. 1044–1049 (Apr. , 1986).
28. P. Heckbert (ed.), "Graphic Gems IV" (Academic Press, 1994).
29. M. Herman, "Fast, three-dimensional, collision-free motion planning" in *IEEE Proc. Int. Conf. Robotics Automat.* 2 pp. 1056–1063 (Apr. , 1986).
30. J. E. Hopcroft, J. T. Schwartz and M. Sharir, "Efficient detection of intersections among spheres" (Tech. Rep. 59, Dept. of Computer Science, Courant Inst. of Math. Sciences. , N.Y.University) (Feb. , 1983).
31. P. M. Hubbard, "Interactive collision detection" in *Proc. IEEE Symp. on Research Frontiers in Virtual Reality* 1 pp. 24–31 (Oct. , 1993).
32. P. Jiménez and C. Torras, "Speeding Up Interference Detection Between Polyhedra" in *IEEE Proc. Int. Conf. Robotics Automat.* pp. 1485–1492 (Minneapolis (MN) (Apr. , 1996).
33. P. Jiménez and C. Torras, "Collision detection: a geometric approach" in *Modelling and Planning for Sensor Based Intelligent Robot Systems* (H. Bunke, H. Noltemeier, T. Kanade (eds.)) World Scientific Pub. Co. (Nov. , 1995).
34. J-C. Latombe, "Robot Motion Planning" Kluwer Academic Publishers (SECS 0124, Boston/Dordrecht/London) (1991).
35. M. C. Lin, "Efficient Collision Detection for Animation and Robotics" (PhD Thesis, University of California, Berkeley) (1993).
36. M. C. Lin and J. F. Canny, "An efficient algorithm for incremental distance computation" (to appear in IEEE Trans. Robotics Automat.).

37. M. C. Lin and J. F. Canny, "A fast algorithm for incremental distance calculation" in *IEEE Proc. Int. Conf. Robotics Automat.* 2 (Sacramento (CA)) pp. 1008–1014 (Apr. , 1991).
38. M. C. Lin, D. Manocha and J. F. Canny, "Fast Contact Determination in Dynamic Environments" in *IEEE Proc. Int. Conf. Robotics Automat.* 1 (San Diego (CA)) pp. 602–608 (May, 1994).
39. Y-H. Liu, S. Arimoto and H. Noborio, "A new solid model HSM and its application to interference detection between moving objects" in *J. Robotic Sys.* 8 (1) pp. 39–54 (1991).
40. T. Lozano-Pérez, "Spatial planning: a configuration space approach" in *IEEE Trans. Comput.* 32 (2) pp. 108–120 (Feb. , 1983).
41. T. Lozano-Pérez, "A simple motion-planning algorithm for general robot manipulators" in *IEEE J. Robotics Automat.* 3 (3) pp. 224–238 (June, 1987).
42. T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles" in *Comm. ACM* 22 (10) pp. 560–570 (Oct. , 1979).
43. V. J. Lumelsky, "Effect of kinematics on motion planning for planar robot arms amidst unknown obstacles" in *IEEE J. Robotics Automat.* 3 (3) pp. 207–223 (June, 1987).
44. N. Megiddo and A. Tamir, "Linear time algorithms for some separable quadratic programming problems" in *Operations Research Letters* 13 (4) pp. 203–211 (1993).
45. K. Mehlhorn and K. Simon, "Intersecting two polyhedra one of which is convex" in *Fundamentals of Computation Theory 85, Lecture Notes in Computer Science* 199 pp. 534–542 (1985).
46. W. Meyer, "Distance between boxes: applications to collision detection and clipping" in *IEEE Proc. Int. Conf. Robotics Automat.* 1 (San Francisco (CA)) pp. 597–602 (Apr. , 1986).
47. B. Mirtich and J. F. Canny, "Impulse-based dynamic simulation" in *Tech. Rep.* CSD-94-815 (University of California) (1994).
48. M. Overmars, "Point location in fat subdivisions" in *Information Processing letters* 44 pp. 261–265 (1992).
49. M. Pellegrini, "Stabbing and ray shooting in 3-space" in *Proceedings of the 6thACM Symposium on Computational Geometry* pp. 177–186 (1990).
50. A. P. del-Pobil, M. A. Serna and J. Llovet, "A new representation for collision avoidance and detection" in *IEEE Proc. Int. Conf. Robotics Automat.* 1 (Nice (France)) pp. 246–251 (May, 1992).
51. M. K. Ponamgi, D. Manocha and M. C. Lin, "Incremental algorithms for collision detection between solid models" in *Proceedings of ACM/Siggraph Symposium on Solid Modelling* 1 pp. 293–304 (1995).
52. S. Quinlan, "Efficient Distance Computation between Non-Convex Objects " in *IEEE Proc. Int. Conf. Robotics Automat.* 4 (San Diego (CA)) pp. 3324–3329 (1994).
53. N. K. Sancheti and S. S. Keerthi, "Computation of certain measures of proximity between convex polytopes: a complexity viewpoint" in *IEEE Proc. Int. Conf. Robotics Automat.* 3 (Nice (France)) pp. 2508–2513 (May, 1992).

54. E. Schömer and C. Thiel, "Efficient collision detection for moving polyhedra" (Tech. Rep. MPI-1-94-147, Max Plank Inst. Informatik Saarbr.) (1995).
55. K. Sun and V. Lumelsky, "Path planning among unknown obstacles: the case of a three dimensional cartesian arm" in *IEEE Trans. Robotics Automat.* 8 (6) pp. 776–786 (Dec. , 1992).
56. W. C. Thibault and B. F. Naylor, "Set operations on polyhedra using binary space partitioning trees" in *ACM Computer Graphics* 21 (4) (July, 1987).
57. F. Thomas, "An approach to the movers problem that combines oriented matroid theory and algebraic geometry." in *IEEE Proc. Int. Conf. Robotics Automat.* 1 (Nagoya (J)) (May, 1995).
58. F. Thomas and C. Torras, "Interference detection between non-convex polyhedra revisited with a practical aim" in *IEEE Proc. Int. Conf. Robotics Automat.* 1 (San Diego (CA)) pp. 587–594 (May, 1994).
59. J. Tornero, J. Hamlin and R. B. Kelley, "Spherical-object representation and fast distance computation for robotic applications" in *IEEE Proc. Int. Conf. Robotics Automat.* 2 (Sacramento (CA)) pp. 1602–1608 (Apr. , 1991).
60. G. Turk, "Interactive collision detection for molecular graphics" (Master Thesis, University of North Carolina) (1989).
61. G. Vanecek, "Back-face culling applied to collision detection of polyhedra" (to appear in "Journal of Visualization and Computer Animation").
62. S. Zeghloul, P. Rambeaud and J. P. Lallemand, "A fast distance calculation between convex objects by optimization approach" in *IEEE Proc. Int. Conf. Robotics Automat.* 3 (Nice (France)) pp. 2520–2525 (May, 1992).

Lecture Notes in Control and Information Sciences

Edited by M. Thoma

1993–1997 Published Titles:

Vol. 186: Sreenath, N.
Systems Representation of Global Climate Change Models. Foundation for a Systems Science Approach.
288 pp. 1993 [3-540-19824-5]

Vol. 187: Morecki, A.; Bianchi, G.; Jaworeck, K. (Eds)
RoManSy 9: Proceedings of the Ninth CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators.
476 pp. 1993 [3-540-19834-2]

Vol. 188: Naidu, D. Subbaram
Aeroassisted Orbital Transfer: Guidance and Control Strategies
192 pp. 1993 [3-540-19819-9]

Vol. 189: Ilchmann, A.
Non-Identifier-Based High-Gain Adaptive Control
220 pp. 1993 [3-540-19845-8]

Vol. 190: Chatila, R.; Hirzinger, G. (Eds)
Experimental Robotics II: The 2nd International Symposium, Toulouse, France, June 25-27 1991
580 pp. 1993 [3-540-19851-2]

Vol. 191: Blondel, V.
Simultaneous Stabilization of Linear Systems
212 pp. 1993 [3-540-19862-8]

Vol. 192: Smith, R.S.; Dahleh, M. (Eds)
The Modeling of Uncertainty in Control Systems
412 pp. 1993 [3-540-19870-9]

Vol. 193: Zinober, A.S.I. (Ed.)
Variable Structure and Lyapunov Control
428 pp. 1993 [3-540-19869-5]

Vol. 194: Cao, Xi-Ren
Realization Probabilities: The Dynamics of Queuing Systems
336 pp. 1993 [3-540-19872-5]

Vol. 195: Liu, D.; Michel, A.N.
Dynamical Systems with Saturation Nonlinearities: Analysis and Design
212 pp. 1994 [3-540-19888-1]

Vol. 196: Battilotti, S.
Noninteracting Control with Stability for Nonlinear Systems
196 pp. 1994 [3-540-19891-1]

Vol. 197: Henry, J.; Yvon, J.P. (Eds)
System Modelling and Optimization
975 pp approx. 1994 [3-540-19893-8]

Vol. 198: Winter, H.; Nüßer, H.-G. (Eds)
Advanced Technologies for Air Traffic Flow Management
225 pp approx. 1994 [3-540-19895-4]

Vol. 199: Cohen, G.; Quadrat, J.-P. (Eds)
11th International Conference on Analysis and Optimization of Systems – Discrete Event Systems: Sophia-Antipolis, June 15–16–17, 1994
648 pp. 1994 [3-540-19896-2]

Vol. 200: Yoshikawa, T.; Miyazaki, F. (Eds)
Experimental Robotics III: The 3rd International Symposium, Kyoto, Japan, October 28-30, 1993
624 pp. 1994 [3-540-19905-5]

Vol. 201: Kogan, J.
Robust Stability and Convexity
192 pp. 1994 [3-540-19919-5]

Vol. 202: Francis, B.A.; Tannenbaum, A.R. (Eds)
Feedback Control, Nonlinear Systems, and Complexity
288 pp. 1995 [3-540-19943-8]

Vol. 203: Popkov, Y.S.
Macrosystems Theory and its Applications: Equilibrium Models
344 pp. 1995 [3-540-19955-1]

Vol. 204: Takahashi, S.; Takahara, Y.
Logical Approach to Systems Theory
192 pp. 1995 [3-540-19956-X]

Vol. 205: Kotta, U.
Inversion Method in the Discrete-time Nonlinear Control Systems Synthesis Problems
168 pp. 1995 [3-540-19966-7]

Vol. 206: Aganovic, Z.;.Gajic, Z.
Linear Optimal Control of Bilinear Systems with Applications to Singular Perturbations and Weak Coupling
133 pp. 1995 [3-540-19976-4]

Vol. 207: Gabasov, R.; Kirillova, F.M.; Prischepova, S.V.
Optimal Feedback Control
224 pp. 1995 [3-540-19991-8]

Vol. 208: Khalil, H.K.; Chow, J.H.; Ioannou, P.A. (Eds)
Proceedings of Workshop on Advances inControl and its Applications
300 pp. 1995 [3-540-19993-4]

Vol. 209: Foias, C.; Özbay, H.; Tannenbaum, A.
Robust Control of Infinite Dimensional Systems: Frequency Domain Methods
230 pp. 1995 [3-540-19994-2]

Vol. 210: De Wilde, P.
Neural Network Models: An Analysis
164 pp. 1996 [3-540-19995-0]

Vol. 211: Gawronski, W.
Balanced Control of Flexible Structures
280 pp. 1996 [3-540-76017-2]

Vol. 212: Sanchez, A.
Formal Specification and Synthesis of Procedural Controllers for Process Systems
248 pp. 1996 [3-540-76021-0]

Vol. 213: Patra, A.; Rao, G.P.
General Hybrid Orthogonal Functions and their Applications in Systems and Control
144 pp. 1996 [3-540-76039-3]

Vol. 214: Yin, G.; Zhang, Q. (Eds)
Recent Advances in Control and Optimization of Manufacturing Systems
240 pp. 1996 [3-540-76055-5]

Vol. 215: Bonivento, C.; Marro, G.; Zanasi, R. (Eds)
Colloquium on Automatic Control
240 pp. 1996 [3-540-76060-1]

Vol. 216: Kulhavý, R.
Recursive Nonlinear Estimation: A Geometric Approach
244 pp. 1996 [3-540-76063-6]

Vol. 217: Garofalo, F.; Glielmo, L. (Eds)
Robust Control via Variable Structure and Lyapunov Techniques
336 pp. 1996 [3-540-76067-9]

Vol. 218: van der Schaft, A.
L_2 Gain and Passivity Techniques in Nonlinear Control
176 pp. 1996 [3-540-76074-1]

Vol. 219: Berger, M.-O.; Deriche, R.; Herlin, I.; Jaffré, J.; Morel, J.-M. (Eds)
ICAOS '96: 12th International Conference on Analysis and Optimization of Systems - Images, Wavelets and PDEs:
Paris, June 26-28 1996
378 pp. 1996 [3-540-76076-8]

Vol. 220: Brogliato, B.
Nonsmooth Impact Mechanics: Models, Dynamics and Control
420 pp. 1996 [3-540-76079-2]

Vol. 221: Kelkar, A.; Joshi, S.
Control of Nonlinear Multibody Flexible Space Structures
160 pp. 1996 [3-540-76093-8]

Vol. 222: Morse, A.S.
Control Using Logic-Based Switching
288 pp. 1997 [3-540-76097-0]

Vol. 223: Khatib, O.; Salisbury, J.K.
Experimental Robotics IV: The 4th International Symposium, Stanford, California, June 30 - July 2, 1995
596 pp. 1997 [3-540-76133-0]

Vol. 224: Magni, J.-F.; Bennani, S.; Terlouw, J. (Eds)
Robust Flight Control: A Design Challenge
664 pp. 1997 [3-540-76151-9]

Vol. 225: Poznyak, A.S.; Najim, K.
Learning Automata and Stochastic Optimization
219 pp. 1997 [3-540-76154-3]

Vol. 226: Cooperman, G.; Michler, G.; Vinck, H. (Eds)
Workshop on High Performance Computing and Gigabit Local Area Networks
248 pp. 1997 [3-540-76169-1]

Vol. 227: Tarbouriech, S.; Garcia, G. (Eds)
Control of Uncertain Systems with Bounded Inputs
203 pp. 1997 [3-540-76183-7]

Vol. 228: Dugard, L.; Verriest, E.I. (Eds)
Stability and Control of Time-delay Systems
337 pp. 1998 [3-540-76193-4]