Motion Planning for Nonholonomic Vehicles: An Introduction

Bill Triggs
Oxford University Robotics Group,
Department of Engineering Science,
Parks Rd, Oxford OX1 3PJ, U.K.
bill@robots.oxford.ac.uk

Abstract

The last few years have seen a burst of activity in motion planning for vehicles with rolling constraints such as cars and trucks with trailers. Our understanding of such **nonholonomic** constraints has increased significantly and it is now possible to quickly plan complex parking and trailer-backing manoeuvres for these practically important systems. However the techniques used are highly mathematical and have not been easily accessible to a broad robotics-oriented audience. This paper presents an intuitive, geometrical introduction to the main mathematical definitions and theorems and summarizes some of the key practical approaches to nonholonomic vehicle planning.

Contents

1 Introduction

There has recently been significant progress in motion planning for car-like and articulated mobile robots. Such vehicles have many potential applications, but planning for them is difficult because they are subject to rolling constraints that limit the possible directions of motion: they can not move sideways directly, but must move forwards or backwards in order to turn. Hence, complicated manoeuvres may be required to move from one configuration to another nearby one, even in the absence of obstacles. Rudder steered vehicles like ships and aeroplanes are subject to similar constraints. Such constraints, that limit the possible directions of motion at a point but can be 'undone' by local manoeuvring, are called **nonholonomic**.

During the last 5 years, several workers have combined conventional robot planning techniques with mathematical results on nonholonomic control obtained over the last few decades, to produce effective planners for nonholonomic vehicles [29]. These are likely to see practical application soon, in systems for AGV parking and docking and controllers for backing articulated vehicles. They are also one step along the road from 'simple' 'piano movers' or 'sliding polygon' algorithms to full kinodynamic planners capable of dealing optimally with complex combinations of kinematic, dynamic and scheduling constraints.

There are several other applications of nonholonomy to robotics, which have been very important in the development of the theory. Murray and Sastry [31] have developed an elegant theory of manipulation with rolling constraints and applied it to multifingered grasping. Nonholonomic algebra also applies to the angular momentum manipulations used to control the orientation of freely falling bodies such as space robots and falling cats [15, 34].

In a broader context, the notion of nonholonomy is deeply entwined with the mathematics of noncommutative symmetries such as the 3D rotation group and the 'gauge' symmetries of general relativity and elementary particle theory. In fact a physical theory is 'gauged' precisely by allowing its local nonholonomy ('gauge curvature') to become a non-trivial, dynamic quantity. Much of the strong development of the theory is a direct consequence of these close ties with fundamental mathematics and physics.

This paper presents a simple, geometric, tutorial introduction to some of the key definitions and theorems of nonholonomy, followed by short descriptions of several recent nonholonomic vehicle planners. Most technical details and mathematical niceties will be ignored, although they are often indispensable to a deeper understanding of the subject. The book by Li and Canny [29] is a good starting place for further reading.

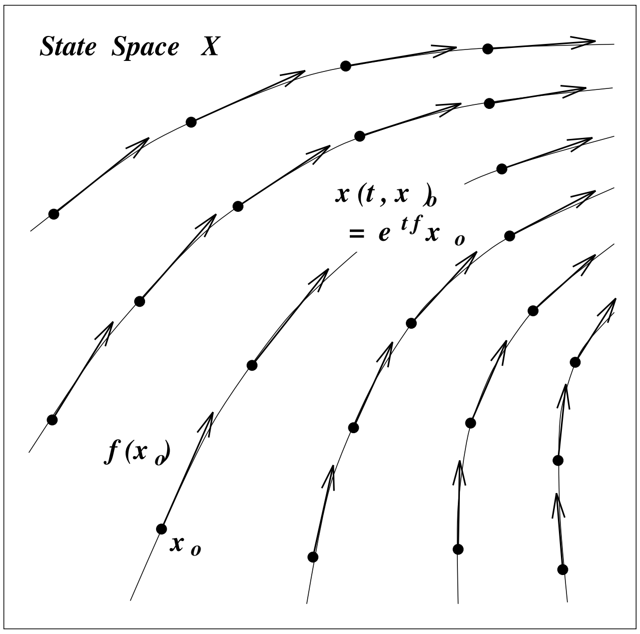


Figure 1: System dynamics is described by a vector field in state space.

2 Dynamical Systems

2.1 State Space and Vector Fields

This section summarizes some standard dynamical systems terminology. We will work in a **state space** $\boldsymbol{X}$ that specifies the vehicle configuration and enough velocities, accelerations, and so forth to give **first order dynamical equations**:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$$

for the state-trajectory $\boldsymbol{x} \equiv \boldsymbol{x}(t)$ of the system. When dynamical quantities can be ignored (for example if we are planning feasible paths rather than optimal trajectories), the state space will often be configuration space. Otherwise it will usually contain velocity as well as configuration dimensions.

Robot state spaces are typically differentiable manifolds with boundaries, but for most of our purposes they can be pictured as pieces of $\boldsymbol{R}^n$ for some n. Differentiable manifolds have **tangent spaces**: at each point $\boldsymbol{x}$ there is a linear space $\boldsymbol{T_x X}$ of **tangent vectors** or possible directions of motion in the manifold. A dynamical function $\boldsymbol{f}$ specifies a **vector field** on state space, i.e. a smooth choice of a tangent vector $\boldsymbol{f}(\boldsymbol{x})$ at each point of $\boldsymbol{X}$. The dynamical law $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ fixes the state velocity $\dot{\boldsymbol{x}}$ to be the vector $\boldsymbol{f}(\boldsymbol{x})$ at each point.

A vector field can be pictured as a field of arrows, one attached to each point of the manifold (fig. 1).

Any vector field $\boldsymbol{f}$ on $\boldsymbol{X}$ can be **integrated** to give a **flow** $\boldsymbol{x}(t, \boldsymbol{x}_0) : \boldsymbol{R} \times \boldsymbol{X} \to \boldsymbol{X}$. For each $\boldsymbol{x}_0$ in $\boldsymbol{X}$ this specifies the **trajectory** $\boldsymbol{x}(t, \boldsymbol{x}_0)$ through $\boldsymbol{x}_0$ satisfying:

$$\begin{aligned} \dot{\boldsymbol{x}}(t, \boldsymbol{x}_0) &= \boldsymbol{f}(\boldsymbol{x}(t, \boldsymbol{x}_0)) \\ \boldsymbol{x}(0, \boldsymbol{x}_0) &= \boldsymbol{x}_0 \end{aligned}$$

The existence theorems for first order ordinary differential equations (ODE's) show that flows always exist locally for smooth enough $\boldsymbol{f}$.

Vector fields can be viewed as operators that generate infinitesimal displacements, in which case integration formally corresponds to operator exponentiation (defined by an operator power series as for matrix exponentiation: $\mathrm{e}^{t\boldsymbol{f}} \equiv \sum_{n=0}^{\infty} \frac{(t\boldsymbol{f})^n}{n!}$). Two realisations of this idea are important for a vector field $\boldsymbol{f}$ on a space $\boldsymbol{X}$:

1. $\boldsymbol{f}$ generates infinitesimal mappings $\boldsymbol{x} \to \boldsymbol{x} + t\boldsymbol{f}$ of $\boldsymbol{X}$ onto itself ($t \ll 1$). The exponential $\mathrm{e}^{t\boldsymbol{f}} : \boldsymbol{X} \to \boldsymbol{X}$ represents the operation of moving along (the integral curves of) $\boldsymbol{f}$ for time t:

$$\begin{aligned} \mathrm{e}^{t\boldsymbol{f}} \cdot \boldsymbol{x}_0 &= \boldsymbol{x}(t, \boldsymbol{x}_0) \\ \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{e}^{t\boldsymbol{f}} \cdot \boldsymbol{x} &= \boldsymbol{f}(\mathrm{e}^{t\boldsymbol{f}} \cdot \boldsymbol{x}) \end{aligned}$$

   We will often use this exponential notation to denote the application of a dynamical law to the system for a finite interval of time.

2. $\boldsymbol{f}$ generates infinitesimal translations of the smooth functions on $\boldsymbol{X}$ by $\phi(\boldsymbol{x}) \to \phi(\boldsymbol{x} + t\boldsymbol{f}) \approx \phi(\boldsymbol{x}) + t\boldsymbol{f} \cdot \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\phi(\boldsymbol{x})$. Then $\mathrm{e}^{t\boldsymbol{f}} \equiv \mathrm{e}^{t\boldsymbol{f} \cdot \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}}$ **pulls back** ϕ from $\boldsymbol{x}(t, \boldsymbol{x}_0)$ to $\boldsymbol{x}_0$: $(\mathrm{e}^{t\boldsymbol{f}} \cdot \phi)(\boldsymbol{x}_0) = \phi(\boldsymbol{x}(t, \boldsymbol{x}_0))$.

2.2 Control Laws

Now assume the system has some **controls** $\boldsymbol{u}$ (steering, accelerator,...) which alter the dynamics:

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

Each setting of the controls gives a different vector field or dynamical law $\boldsymbol{f}(-, \boldsymbol{u})$. We can 'drive' the system to different regions of state space by manipulating $\boldsymbol{u}$. For example a **feedback control law** $\boldsymbol{u} = \boldsymbol{u}(t, \boldsymbol{x})$ tells the vehicle controller how to respond at each time at each point of state space. Such laws generate an implicit closed loop dynamics $\dot{\boldsymbol{x}} = \tilde{\boldsymbol{f}}(t, \boldsymbol{x})$, and may be chosen to achieve some given task such as path following or parking.

Sometimes, controls can be defined so that the system is linear in $\boldsymbol{u}$:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_0(\boldsymbol{x}) + \sum_i \boldsymbol{f}_i(\boldsymbol{x}) u^i$$

In fact, all of the systems we will consider can be made control-linear and **drift free** $\boldsymbol{f}_0 \equiv 0$. The $\boldsymbol{f}_i$ are called **control vector fields**. For a linear system the possible directions of motion at a point are linear combinations of the control fields, so the state velocity is implicitly constrained to lie in $\mathrm{Span}\{\boldsymbol{f}_i\}$.

Even if the system is not linear, by rapidly alternating between several different control vectors we can effectively apply any convex linear combination of them, so there is still a kind of residual linearity in $\boldsymbol{u}$. In fact it turns out that many aspects of non-linear control are well captured by the linear theory [38].

3 Types of Constraint

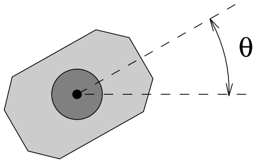
Several types of constraint are important for vehicles. Planners that consider a wider range of constraints can usually produce better plans, but there is often a high cost in computational and code complexity.

Configuration constraints restrict the possible static configurations of the vehicle by delimiting forbidden regions of state or configuration space called **configuration space obstacles (CSO's)**. The main types are **workspace obstacles** (physical objects the vehicle could collide with) and **joint limits** (eg bounds on steering lock or possible tractor-trailer angles).

Dynamical constraints restrict the values of differential quantities such as velocity, acceleration and path curvature. If the vehicle state contains the relevant velocity, ..., components they translate into forbidden regions of state space. Otherwise they limit the allowable state space velocities and control laws. **Dynamical bounds** are inequalities, for example on the maximum permissible speed or path curvature. **Differential constraints** are equalities which restrict the velocity to a submanifold of the possible state space directions, for example railway tracks and rolling constraints. The main classes of

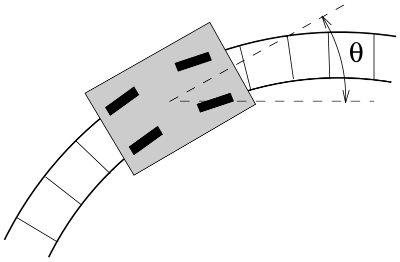
Holonomic Systems

Hovercraft



$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} v_x \\ v_y \\ v_\theta \end{pmatrix}$$

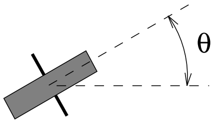
Train



$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ c(x,y) \end{pmatrix}(v)$$

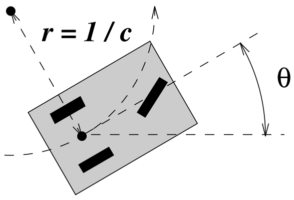
Nonholonomic Systems

Unicycle



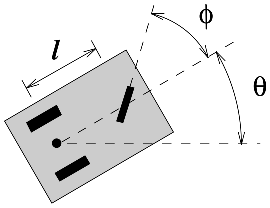
$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} v \\ v_\theta \end{pmatrix}$$

Car with fast steering



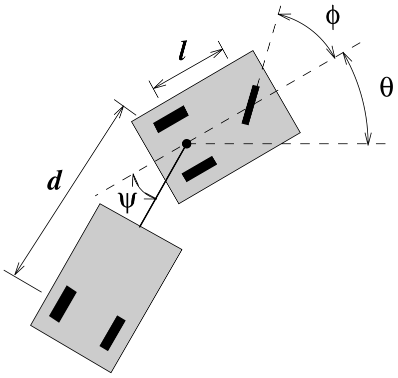
$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} v \\ v\,c \end{pmatrix}$$

Car with slow steering



$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix} x \\ y \\ \theta \\ \phi \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ \tan(\phi)/l & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} v \\ v_\phi \end{pmatrix}$$

Car with trailer



$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix} x \\ y \\ \theta \\ \psi \\ \phi \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ \tan(\phi)/l & 0 \\ \sin(\psi)/d & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} v \\ v_\phi \end{pmatrix}$$

Figure 2: Some examples of holonomic and nonholonomic systems.

differential constraints are **holonomic** or integrable and **nonholonomic** or nonintegrable constraints.

Integral constraints do not translate directly into local state space quantities, but limit the values of integrals such as execution time and resource consumption over an entire task. **Scheduling constraints** are particularly significant, especially when there are interactions such as moving objects in the environment that the vehicle must avoid.

Optimal planners minimize the value of some integral cost function subject to combinations of the above constraints, while **heuristic planners** 'merely' produce *some* constraint-satisfying plan.

4 Some Simple Examples

Figure 2 shows some simple examples of holonomic and nonholonomic vehicles. Note that all of them can be expressed as control-linear drift-free systems, so that their possible motions are linear combinations of several control vector fields (the columns of the matrices).

5 Small Manoeuvres and Lie Brackets

How can we characterize the effects of small manoeuvres?

Consider a car trying to parallel park. It can not move sideways directly, but a small local manoeuvre can produce the required sideways motion (fig. 3). Move straight forwards for a distance s, turn outwards for a distance t with curvature c (turn radius $1/c$), reverse straight backwards for s, and reverse and turn for t. By elementary geometry the result is a second order sideways displacement $c\,s\,t + \mathcal{O}(s^2 t, st^2)$.

This is an example of a general scenario. Suppose we have a system with at least two distinct 'forward' control settings and the corresponding 'reverses', giving dynamical laws or state space vector fields $\boldsymbol{f}$, $-\boldsymbol{f}$, $\boldsymbol{g}$, $-\boldsymbol{g}$, ... Now apply a 'closed sequence of infinitesimal controls', i.e. a sequence of controls, each applied for an infinitesimal time, which is balanced in the sense that there is exactly the same amount of $\boldsymbol{f}$ as $-\boldsymbol{f}$, $\boldsymbol{g}$ as $-\boldsymbol{g}$, and so on. For example, $\mathrm{e}^{-t\boldsymbol{g}} \cdot \mathrm{e}^{-s\boldsymbol{f}} \cdot \mathrm{e}^{t\boldsymbol{g}} \cdot \mathrm{e}^{s\boldsymbol{f}}$ is the simplest non-trivial closed sequence (fig. 4). Although this

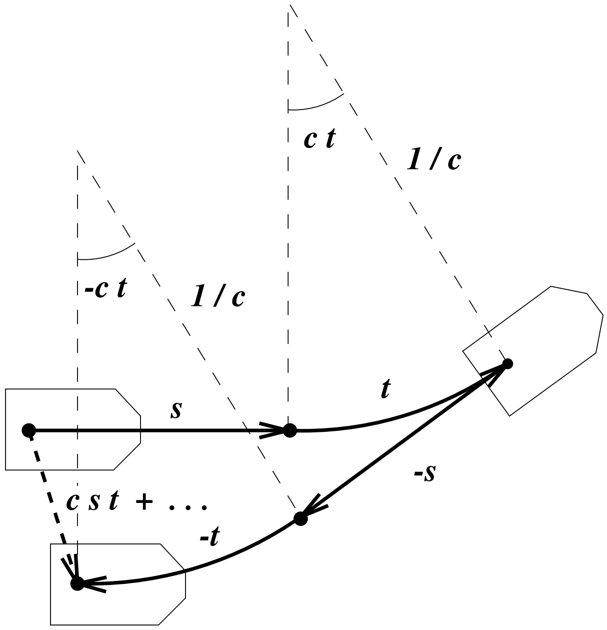


Figure 3: Backwards and forwards motions can produce a sideways displacement that parallel parks a car.

is a 'closed path in control space' in the sense that the total integrated control signal is zero, it does *not* in general produce a closed path in state space. A careful calculation shows that the residual is a second order motion:

$$\mathrm{e}^{-t\boldsymbol{g}} \cdot \mathrm{e}^{-s\boldsymbol{f}} \cdot \mathrm{e}^{t\boldsymbol{g}} \cdot \mathrm{e}^{s\boldsymbol{f}} = \mathrm{e}^{st[\boldsymbol{f},\boldsymbol{g}]+\cdots}$$

where $[\boldsymbol{f},\boldsymbol{g}]^a \equiv \sum_b (\boldsymbol{f}^b \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}^b})\boldsymbol{g}^a - (\boldsymbol{g}^b \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}^b})\boldsymbol{f}^a$, ie:

$$\boxed{[\boldsymbol{f},\boldsymbol{g}] \equiv (\boldsymbol{f} \cdot \nabla)\boldsymbol{g} - (\boldsymbol{g} \cdot \nabla)\boldsymbol{f}}$$

Given any two vector fields $\boldsymbol{f}$ and $\boldsymbol{g}$, the quantity $[\boldsymbol{f},\boldsymbol{g}]$ is called the **Lie Bracket**[1] of $\boldsymbol{f}$ and $\boldsymbol{g}$. It is a geometric (coordinate system independent), skew symmetric, bilinear, differential product on the space of all vector fields, which tells how one vector field changes as you move along the other one.

$$[\boldsymbol{f},\boldsymbol{g}] = -[\boldsymbol{g},\boldsymbol{f}] \qquad \text{skew symmetry}$$
$$[\boldsymbol{f}, s\boldsymbol{g} + t\boldsymbol{h}] = s[\boldsymbol{f},\boldsymbol{g}] + t[\boldsymbol{f},\boldsymbol{h}] \qquad \text{linearity}$$
$$[\boldsymbol{f},[\boldsymbol{g},\boldsymbol{h}]] + [\boldsymbol{g},[\boldsymbol{h},\boldsymbol{f}]] + [\boldsymbol{h},[\boldsymbol{f},\boldsymbol{g}]] = 0 \qquad \text{Jacobi identity}$$
$$\mathrm{e}^{s\boldsymbol{f}} \cdot \mathrm{e}^{t\boldsymbol{g}} = \mathrm{e}^{s\boldsymbol{f}+t\boldsymbol{g}-\frac{1}{2}st[\boldsymbol{f},\boldsymbol{g}]+\cdots}$$
$$\mathrm{e}^{-t\boldsymbol{g}} \cdot \mathrm{e}^{-s\boldsymbol{f}} \cdot \mathrm{e}^{t\boldsymbol{g}} \cdot \mathrm{e}^{s\boldsymbol{f}} = \mathrm{e}^{st[\boldsymbol{f},\boldsymbol{g}]+\cdots}$$

Campbell-Hausdorff-Baker-Dynkin identities

[1] After the great Norwegian mathematician Sophus Lie (pronounced 'lee') 1842-99, the father of the theory of continuous symmetry groups.

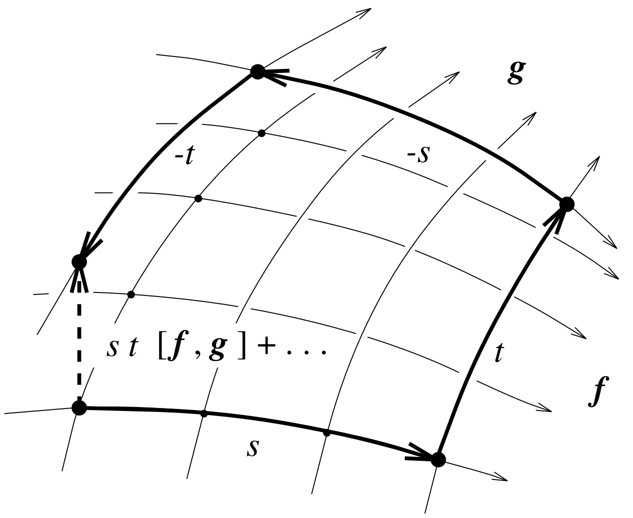


Figure 4: 'Closed' controls can produce a net displacement in state space.

Returning to our manoeuvring car example, we see that the Lie bracket of the two control vector fields is:

$$\left[\begin{pmatrix}\cos\theta\\ \sin\theta\\ c\end{pmatrix},\begin{pmatrix}\cos\theta\\ \sin\theta\\ 0\end{pmatrix}\right]$$

$$= \left(\cos\theta\cdot\tfrac{\mathrm{d}}{\mathrm{d}x}+\sin\theta\cdot\tfrac{\mathrm{d}}{\mathrm{d}y}+c\cdot\tfrac{\mathrm{d}}{\mathrm{d}\theta}\right)\begin{pmatrix}\cos\theta\\ \sin\theta\\ 0\end{pmatrix}$$

$$-\left(\cos\theta\cdot\tfrac{\mathrm{d}}{\mathrm{d}x}+\sin\theta\cdot\tfrac{\mathrm{d}}{\mathrm{d}y}+0\cdot\tfrac{\mathrm{d}}{\mathrm{d}\theta}\right)\begin{pmatrix}\cos\theta\\ \sin\theta\\ c\end{pmatrix}$$

$$= c\cdot\begin{pmatrix}-\sin\theta\\ \cos\theta\\ 0\end{pmatrix}$$

So again we find that there is a sideways displacement by $c\,s\,t + \mathcal{O}(st^2, s^2t)$.

6 Integrability and Nonholonomy

Suppose we have a differential constraint restricting the allowed state velocities to a linear subspace of the possible tangent directions. When are there small manoeuvres which 'undo' this constraint, and in which directions can we move?

6.1 The Frobenius Theorem

The idea of a linear differential constraint can be formalized as follows:

A ***k*-plane distribution**[2] Δ on a manifold $\boldsymbol{X}$ is a smoothly varying choice of a k dimensional linear subspace $\Delta_{\boldsymbol{x}}$ of the tangent space at each $\boldsymbol{x}$ in $\boldsymbol{X}$. A trajectory or vector field $\boldsymbol{f}$ is said to be **in** Δ whenever $\boldsymbol{f}(\boldsymbol{x})$ is in $\Delta_{\boldsymbol{x}}$ at each point.

For our systems $\Delta_{\boldsymbol{x}}$ will be the span of the control vector fields at $\boldsymbol{x}$ (i.e., the possible motion directions at $\boldsymbol{x}$). In fact, any k vector fields or $n-k$ linear constraint equations define a k-plane distribution, so long as they are linearly independent at each point. However distributions are geometric objects that may exist without reference to any particular choice of vector fields or constraints.

The key theorem for distributions is the **Frobenius theorem**[3]:
The constraint Δ can be 'integrated' so that $\boldsymbol{X}$ is filled ('foliated') by an $n-k$ dimensional family of k dimensional submanifolds with tangent planes in Δ if and only if Δ is closed under the Lie bracket (i.e., for all $\boldsymbol{f},\boldsymbol{g}$ in Δ, $[\boldsymbol{f},\boldsymbol{g}]$ is in Δ). The idea is that locally we can try to join up the tangent planes $\Delta_{\boldsymbol{x}}$ into pieces of submanifold, but the pieces will fail to 'mesh' if small manoeuvres produce new displacements that lie outside $\Delta_{\boldsymbol{x}}$, since going around an infinitesimal loop would allow us to move out of the surface. So the obstruction to forming a submanifold is precisely the extent to which $[\Delta,\Delta]$ is not contained in Δ.

Notes:

1. The Frobenius theorem is always true for 1 dimensional (line) distributions, which are essentially ODE's without a choice of time scaling.
2. For practical calculations it suffices to test for the closure of Δ on any k independent vector fields in Δ: check whether $[\boldsymbol{f}_i,\boldsymbol{f}_j](\boldsymbol{x})$ is expressible as $\sum_l c^l_{ij}(\boldsymbol{x})\boldsymbol{f}_l(\boldsymbol{x})$ for some scalar functions c^l_{ij} and all $i,j=1,\dots,k$.
3. The Lie bracket can also be used to tell whether a set of independent vector fields $\boldsymbol{f}_1,\dots,\boldsymbol{f}_k$ closed under $[\,,]$ define *coordinate systems* for their submanifolds. Locally, by moving λ_1 along $\boldsymbol{f}_1$ then λ_2 along $\boldsymbol{f}_2$ then ...then λ_k along $\boldsymbol{f}_k$ we can get to any nearby point of the submanifold, so we might try to use $(\lambda_1,\dots,\lambda_k)$ as local coordinates. But we would usually get

[2] This has nothing to do with *functional* distributions such as the Dirac δ-function.

[3] A reformulation of a theorem by the German group theorist Georg Frobenius, 1849-1917.

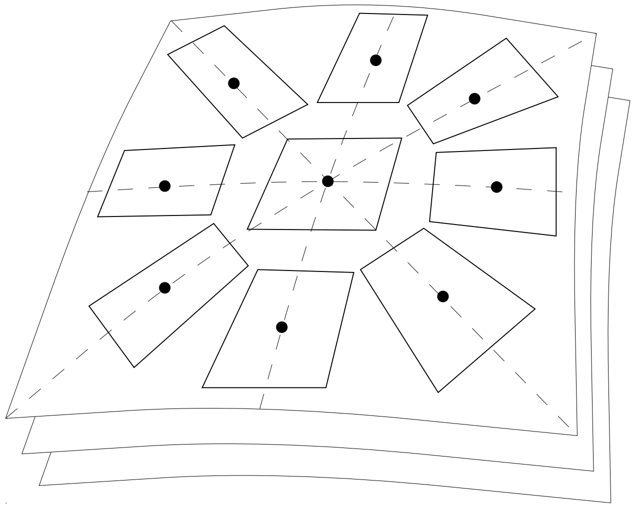


Figure 5: Holonomic constraints mesh together into integral submanifolds that the system can not leave.

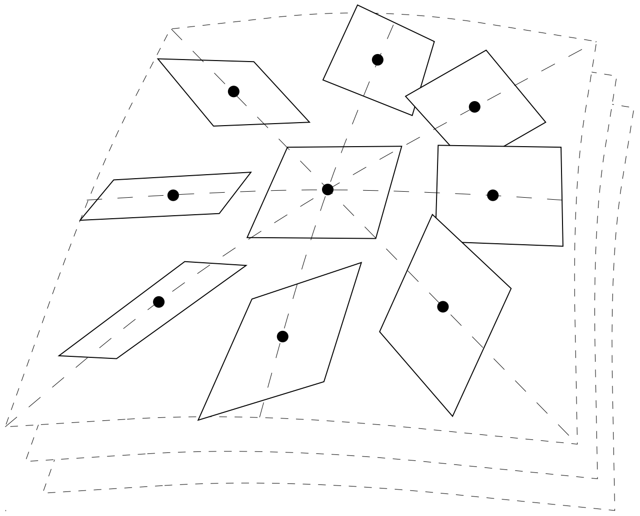


Figure 6: Nonholonomic constraints 'twist out of the constraint plane' and fail to mesh into submanifolds, so small manoeuvres can move the system across the constraints.

different λ's if we followed the $\boldsymbol{f}$'s in a different order. It turns out that the integral curves 'mesh' to form a coordinate system if and only if $[\boldsymbol{f}_i, \boldsymbol{f}_j] \equiv 0$ for all i, j.

4. There is a very nice dual form of the Frobenius theorem for differential forms that can be used to test the integrability of distributions specified by $n-k$ linear constraints $\boldsymbol{h}^1, \ldots, \boldsymbol{h}^{n-k}$ (where $\boldsymbol{f}$ is in Δ if and only if $\boldsymbol{h}^i \cdot \boldsymbol{f} = 0$ for $i = 1, \ldots, n-k$). Given constraint forms $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ there are geometric operations called the **exterior product** $\wedge$ and the **exterior derivative d** that generalize the cross product and curl operations on 3-vectors:

$$(\boldsymbol{g} \wedge \boldsymbol{h})_{ab} = \frac{1}{2}(g_a h_b - g_b h_a)$$
$$(\mathbf{d}\boldsymbol{g})_{ab} \equiv \frac{1}{2}(\frac{\partial g_b}{\partial x^a} - \frac{\partial g_a}{\partial x^b})$$

Both produce skew symmetric matrices and both generalize to all ***k*-forms** or completely antisymmetric k-index tensor fields $\boldsymbol{g}_{ab\cdots d}(\boldsymbol{x})$ (where $\boldsymbol{g}_{a\cdots b\cdots c\cdots d} = -\boldsymbol{g}_{a\cdots c\cdots b\cdots d}$ for all choices of b and c). The Frobenius integrability condition can then be written $\mathbf{d}\boldsymbol{h}^i(\boldsymbol{x}) = \sum_j \boldsymbol{c}^{ij}(\boldsymbol{x}) \wedge \boldsymbol{h}^j(\boldsymbol{x})$ for some set of coefficient forms $c_a^{ij}(\boldsymbol{x})$. In other words the exterior derivative of each of the constraint forms must be expressible as a linear combination (with form-valued coefficients) of the other constraint forms. In a sense, the Lie bracket is 'dual' to the exterior derivative **d**.

6.2 Holonomic and Nonholonomic Constraints

If the Frobenius theorem holds, the state space looks rather like an onion, with many distinct 'skins' or layers (fig. 5). The dynamical equations force the system to stay in the layer it starts in, so the rest of the space is essentially irrelevant to it. In this case the state space can be '**reduced**' (separated into disjoint submanifolds) and the constraints are '**holonomic**'.

If the constraint distribution Δ is not closed under the Lie bracket the constraints are '**nonholonomic**'. In this case the constraint spaces 'twist up out of the plane like a fan blade' and fail to mesh to form submanifolds, so that small manoeuvres can 'undo' the constraints (fig. 6).

Figure 7 shows a more detailed picture of the constraint distribution for a car-like vehicle whose state is specified by its location and orientation in the plane. At each point the car must move in the direction of its heading, but it can alter its speed and rate of turn (the 'horizontal' and 'vertical' components of state velocity in the diagram, representing the two columns of the control field matrix in fig. 2). A circular path appears as a constant-pitch spiral in state space.

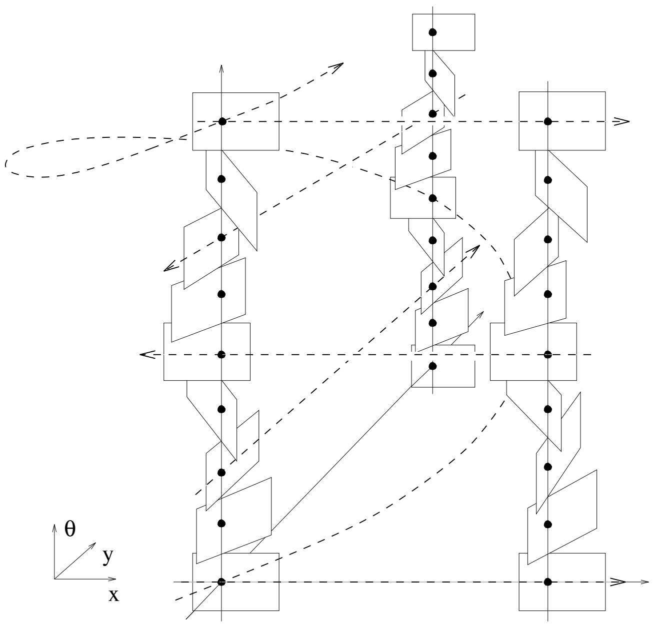


Figure 7: The constraint distribution and some possible configuration space paths for a car. Orientation is plotted vertically.

7 Parking Isn't Easy

Nonholonomic constraints can make it difficult to correct small errors and perturbations. For example, if you are parking your car in a garage and think you are going to end up $\mathcal{O}(\delta)$ too close to the side wall, you will need a forward distance $\mathcal{O}(\sqrt{\delta})$ to make the sideways correction at full steering lock. The closer you are to the stopping point, the harder it is to make adjustments.

Appendix A outlines a result by Brockett [8] that formalizes this difficulty. It says roughly that there is *no* continuous asymptotically stable control law $\boldsymbol{u} \equiv \boldsymbol{u}(\boldsymbol{x})$ that will park a nonholonomic system at a point, so small perturbations *can not* be asymptotically corrected by any choice of state-dependent feedback.

Note that the theorem is only valid for parking at a point. Stabilising feedback laws for nonholonomic *path following* are certainly possible, and Coron [12] has shown that even a point can be stabilized with a *time-varying* (oscillatory) feedback law $\boldsymbol{u} = \boldsymbol{u}(t, \boldsymbol{x})$ for which side-slipping reversals are possible.

8 Local Controllability

We have seen that small local manoeuvres can 'undo' a nonholonomic constraint to second order along directions $[\boldsymbol{f}, \boldsymbol{g}]$. Repeating the process for $\boldsymbol{f}$ and $[\boldsymbol{g}, \boldsymbol{h}]$ we can generate a third order displacement along $[\boldsymbol{f}, [\boldsymbol{g}, \boldsymbol{h}]]$, and so on. How far can we go?

8.1 The Controllability Rank Condition

Given the distribution Δ spanned by the controls, define the **Control Lie Algebra** $\mathcal{LA}(\Delta)$ as the closure of Δ under the Lie bracket, i.e.:

$$\begin{aligned} \Delta_1 &\equiv \Delta \\ \Delta_k &\equiv \Delta_{k-1} \oplus [\Delta, \Delta_{k-1}] \\ \mathcal{LA}(\Delta) &\equiv \bigcup_k \Delta_k \end{aligned}$$

$\mathcal{LA}(\Delta)_{\boldsymbol{x}}$ contains all of the directions spanned by all of the Lie products of vector fields in Δ at $\boldsymbol{x}$. Local manoeuvres of high enough order can move the system along any of these directions at $\boldsymbol{x}$.

W. L. Chow's '**local controllability rank condition**' [11] tells us when we can move a small but finite distance in any direction:
If $\mathcal{LA}(\Delta)_{\boldsymbol{x}}$ has maximum rank $(\dim(\boldsymbol{X}))$ at $\boldsymbol{x}$ in $\boldsymbol{X}$, there is an open neighbourhood $\mathcal{N}$ of $\boldsymbol{x}$ in which each point is reachable from $\boldsymbol{x}$ by a local manoeuvre lying entirely in $\mathcal{N}$. If $\mathrm{Rank}(\mathcal{LA}(\Delta)_{\boldsymbol{x}}) = \dim(\boldsymbol{X})$ almost everywhere the system is said to be **globally controllable**. In this case Chow's theorem shows that a path exists for the nonholonomic system *if and only if* one exists for the corresponding *unconstrained* holonomic system. In fact *any* unconstrained path can be followed arbitrarily closely by the nonholonomic system: you can drive your car sideways if you want to! – The catch is that many many small see-sawing manoeuvres are required to approximate a sideways path closely, so you have to work very hard to do it.

8.2 Testing for Controllability

Testing for local controllability is not easy in general, although all of the vehicles we will consider turn out to be locally controllable almost everywhere, including cars and tractors with arbitrary numbers of trailers [25].

There is heavy algebraic machinery to enumerate all of the possible Lie brackets at a given order (taking skew symmetry and the Jacobi identity into account) and generate 'Philip Hall bases' for $\mathcal{LA}(\Delta)_{\boldsymbol{x}}$ [7, 26]. But the algorithm is exponential in the power of $[\,,\,]$ and *arbitrarily* high powers may be required to generate $\mathcal{LA}(\Delta)_{\boldsymbol{x}}$ at some points. Globally things are even more complicated as Δ typically has 'singular' hypersurfaces where the control fields are linearly dependent, and $\mathcal{LA}(\Delta)_{\boldsymbol{x}}$ may fail to have full rank on these and other hypersurfaces.

9 Synthesising Local Paths

Even when a system is known to be locally controllable, explicit formulas for manoeuvres to generate particular displacements can be hard to find. So the first step in global nonholonomic planning is working out how to move locally.

There are a number of algebraic methods for local path synthesis in special cases, but transforming the system so that they can be applied is something of an art form. The general strategy is to 'fix' successively higher powers of $[\,,\,]$ in turn, but this rather ad hoc approach precludes any form of optimality. Explicit formulas for locally optimal paths are available only for the simplest systems and cost functions. (For example the minimal length paths for limited-steering-lock cars are known: see §13).

9.1 Sinusoidal Controls

We have seen that repetitive see-sawing motions can move a car sideways. To produce these an 'oscillatory' time-varying control is required. Murray and Sastry [32, 33] show how to use piecewise sinusoidal control signals to synthesize paths for any system of the 'triangular' form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_p \end{pmatrix} = \begin{pmatrix} 1 \\ \boldsymbol{f}_2(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{f}_p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p) \end{pmatrix} \cdot \boldsymbol{u}$$

The method applies a series of p sinusoidal controls that zero each $\boldsymbol{x}_i$ in turn. The frequencies and amplitudes must be carefully chosen to control the current $\boldsymbol{x}_i$ while leaving the previous $\boldsymbol{x}_j$'s undisturbed.

Any locally controllable system can implicitly be written in triangular form by an appropriate change of coordinates. The transformation can be made explicit for cars and cars with arbitrary numbers of trailers [33].

Figure 8: Some 'parking' manoeuvres for a car planned using Lafferriere and Sussmann's technique (from [22]).

9.2 Series Expansion in Lie Bracket

Small closed manoeuvres generate displacements along Lie bracket directions, with corrections at higher orders. If we can express a desired displacement as a sum of known Lie brackets, it can be approximately synthesized by applying the manoeuvres that generate the Lie brackets in sequence. However complex higher order correction terms must be accounted for if this is to be made rigorous.

Lafferriere and Sussmann [22] show how to handle these corrections when all the Lie brackets above order k vanish (i.e. $[\boldsymbol{f}_1, [\boldsymbol{f}_2, \ldots [\boldsymbol{f}_k, \boldsymbol{f}_{k+1}] \ldots]] \equiv 0$ for all $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_{k+1}$). Such systems are '**nilpotent of order $\boldsymbol{k}$**'. For other systems the method generates a rapidly convergent sequence of approximate solutions as k is increased.

A rough impressionist sketch of the procedure is as follows:

Choose a set of control fields and brackets $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_r$ spanning the state space and express the desired displacement in terms of them, as $\mathrm{e}^{\sum_{i=1}^{r} \lambda_i \boldsymbol{f}_i}$. Rewrite sums of terms and compound brackets as products, for example:

$$\begin{aligned} \mathrm{e}^{s\boldsymbol{f}+t\boldsymbol{g}} &\rightarrow \mathrm{e}^{-\frac{1}{2}st[\boldsymbol{f},\boldsymbol{g}]+\cdots} \cdot \mathrm{e}^{s\boldsymbol{f}} \cdot \mathrm{e}^{t\boldsymbol{g}} \\ \mathrm{e}^{s[\boldsymbol{f},\boldsymbol{g}]} &\rightarrow \mathrm{e}^{\text{corrections}} \cdot \mathrm{e}^{-\sqrt{s}\boldsymbol{g}} \cdot \mathrm{e}^{-\sqrt{s}\boldsymbol{f}} \cdot \mathrm{e}^{\sqrt{s}\boldsymbol{g}} \cdot \mathrm{e}^{\sqrt{s}\boldsymbol{f}} \end{aligned}$$

Deal with low order terms first, ignore terms of or-

Figure 9: The configurations within 1 unit of path length from the origin for a car (from [27]).

der $> k$, and always be careful to push the remaining corrections towards the end of the product. The result is a finite series of realisable controls which terminates at order k by nilpotency. Lafferriere and Sussmann [22] also show that cars and cars with trailers can be expressed as nilpotent systems by 'feedback nilpotentisation' (a nonlinear, state dependent redefinition of the controls $\boldsymbol{u}$), and present some examples of explicit path calculations. Figure 8 shows some manoeuvres for a front wheel drive car generated by their methods.

10 Path Based Metrics

It is often convenient to have a measure of the cost of travelling between points of state space to provide an estimate of path quality. Nonholonomic systems may require a considerable amount of manoeuvring to travel between nearby configurations, so such cost functions do not correspond well with 'Euclidean distance in state space' (whatever that means). However any positive definite cost function potentially defines a metric and an associated topological structure on state space.

The length of the shortest path between two points is the most obvious choice of cost function. For nonholonomic systems, the balls of this metric (the sets of points within a fixed distance of a given centre) can be very distorted. As their radius shrinks they become ever more squashed and twisted and their surfaces become cusped and fold back on themselves [27, 26] (fig. 9).

However the controllability rank condition (§8.1) implies that the shortest path metric is always topologically equivalent to the Euclidean one at points with full rank. This is comforting: if the topologies were different it might turn out that small perturbations could not be corrected or that apparently well-behaved paths were not in fact continuous. It would certainly mean that many of our basic geometric intuitions about the situation would have to be reconsidered.

11 Optimal and Bang-Bang Control

The above methods generate heuristic paths. It is also possible to plan paths that are in some sense 'optimal'. This section gives an extremely brief introduction to optimal planning in general, and the rest of the paper concentrates on *minimal length* paths for car-like vehicles, for which closed-form results are available.

A broad class of optimal planning problems can be formulated as follows:
Find a control $\boldsymbol{u}(t)$ for $\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u})$ which minimizes the cost $\int_{\boldsymbol{x}_0}^{\boldsymbol{x}_1} c(t, \boldsymbol{x}, \boldsymbol{u})\, dt$ of travelling from $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$, subject to the constraint $\boldsymbol{u}(t) \in \boldsymbol{U}$ for some set of permitted control values U. The key to such **optimal control** problems is the **Pontryagin Maximum Principle** [35]. This is reviewed in appendix B. It reformulates the problem as a Hamiltonian system in an extended state space and uses techniques from classical mechanics to find a useful implicit form for the solutions. The implicit form is sometimes strong enough to allow explicit optimal solutions to be derived, but this usually only happens for fairly simple systems.

The main practical conclusion of the Maximum Principle is that optimal controls are typically hard limited and piecewise continuous, i.e. $\boldsymbol{u}(t)$ is usually hard against the constraint boundaries and switches abruptly between limits at certain critical points. Such controls are called **bang-**

bang controls.

For example, to cycle from Oxford to Cambridge in the shortest possible time you have to: (*i*) accelerate as hard as possible until you reach maximum speed; (*ii*) keep pedalling as fast as possible for as long as you can; (*iii*) brake as hard as possible just in time to stop at the Cambridge. Notice that:

- The 'bangs' divide the problem into natural subtasks.
- The optimal solution is also *locally* optimal in the sense that at each moment you have to do as well as you can for the subtask in hand.
- At each point the optimal control can depend on everything between the current state and the goal, but it is completely independent of the path from the initial state to the current one. So the scheduling of subtasks requires exact foreknowledge but no past knowledge: you have to start braking at exactly the right point no matter how you got there. This potential need for unbounded look ahead makes optimal planning difficult and is one reason why only implicit solutions are easily available.
- Optimal paths are fragile. The controls are at their limits so there is no margin for error: if you fail to brake in time you will overshoot, if you fail to turn as hard as possible you will hit the wall, and so on.

12 Kinodynamic Planning

Kinodynamic planning [9, 13] is the branch of planning dealing with optimal control under complex cost functions and complex kinematic and dynamic constraints. This is a rather broad umbrella, but one common strategy is to discretize time and search either forwards or backwards over all possible sequences of 'bangs' (i.e. controls that are hard against the constraints and constant on each time interval). Since an optimal control schedule typically consists of piecewise continuous hard-limited controls with abrupt transitions, it can usually be well approximated by such a sequence.

Barraquand and Latombe [4] have developed a planner for cars with multiple trailers which uses this technique and $\mathcal{A}^*$ search in an n-dimensional bitmap representing the configuration space. The

Figure 10: Paths for a car (left) and an articulated vehicle (right) planned by the Barraquand-Latombe algorithm (from [4]).

number of reversals is used as an evaluation function, perhaps combined with the path length. The method is relatively simple and resolution complete, and it can handle fairly complex environments. It has time complexity $\mathcal{O}(r^n(r + n \log r))$ at resolution r. This makes it impractical for vehicles with more than one trailer, but it runs quite quickly for cars and cars with one trailer. Figure 10 shows some paths planned by the method.

13 Shortest Paths for Cars

Most nonholonomic systems and cost functions are too complex to allow the derivation of explicit formulae for optimal paths, however analytic forms do exist for the *minimum length* paths of cars with limited steering lock moving between arbitrary configurations in the empty plane.

If the vehicle must always move forwards the shortest paths are known as **Dubins' curves** [14], and if reversals are allowed they are called **Reeds-Shepp curves** [36]. These results form the basis of several effective planners for the practically important case of car-like vehicles. They also shed light on the form of path-length-based metrics and the probable structure of optimal paths for more complex systems. The formulae are complicated but they have the expected bang-bang form, which in this case means that the paths are made up of straight line segments and maximally curved arcs. Figures 11 and 12 show some examples.

Before proceeding, it is important to understand the limitations of the minimum length criterion:

- As with other classes of optimal plans, shortest paths tend to be 'brittle': they often skim the edges of obstacles and there is no margin for execution error as the constraints are kept at their limits (eg, steering is always applied at maximum lock).
- The length-based cost function ignores vehicle dynamics. The shortest paths usually have abrupt curvature discontinuities and direction reversals that require infinitely fast steering and acceleration to track at finite speeds. If acceleration times can not be ignored the shortest paths are *not* usually time optimal. Neither are they optimal under other cost metrics such as efficiency, safety, etc. So what measure of cost do we really want to optimize anyway?
- The form of the optimal paths depends critically on the exact nature of the constraints. For a car-like vehicle with limited speed and steering lock and rapid acceleration the above arc-line form applies. For a vehicle such as the common RoboSoft platform, with two independent drive wheels and independent speed and acceleration limits on each, the shortest paths are combinations of lines, circles, **clothoids** (Euler or Cornu spirals — curvature is an affine function of arc-length) and **anti-clothoids** (involutes of circles) [28, 16]. For a car-like vehicle with limited acceleration and steering slew rate, the steering wheel angle is an affine function of arc-length. For small steering angles this again produces approximately clothoidal path sections.

13.1 Dubins' Curves

Dubins' Theorem [14]: For a car with minimum turning radius R moving forwards in an empty plane, the shortest path between any two configurations is a sequence of straight line segments (S) and circular arcs of radius R (C) of the form CSC, CCC, or a subsequence of one of these. In more detail there are exactly 6 possibilities to check: LSL, LSR, RSL, RSR, LRL, RLR, where $L =$ left turn, $R =$ right turn.

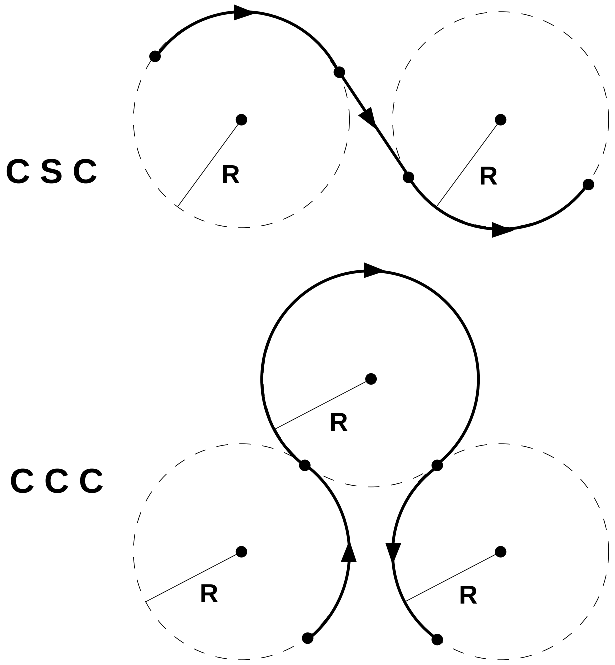


Figure 11: Some examples of Dubins' curves.

13.2 Reeds-Shepp Curves

Reeds-Shepp Theorem [36]: If the car is also allowed to reverse there is *a* shortest path which has the form (a subset of) $C|CSC|C$, where | indicates an optional reversal (cusp). In detail there are 68 possibilities to check. The shortest paths are no longer unique, there may be continuous families of them.

Sketch of proof:

- Use 'elementary' calculus and explicit length estimates to derive path-shortening rewrite rules that convert SCS, SCC and $CCCC$ to Dubins' form and remove cusps from $C|S$, ..., $C|C|C|C$.
- Use the rules to rewrite an arbitrary arc-line path in Reeds-Shepp form by repeatedly reducing sections between cusps to Dubins' form and applying the cusp eliminating rules.
- Given an arbitrary path, uniformly approximate it with a sequence of arc-line paths and reduce each member of the sequence to Reeds-Shepp form. By a compactness argument there is an accumulation point of Reeds-Shepp form. This is the desired shortest path.
- Both results also follow from the Pontryagin Maximum Principle and some analytic calculations [39, 6].

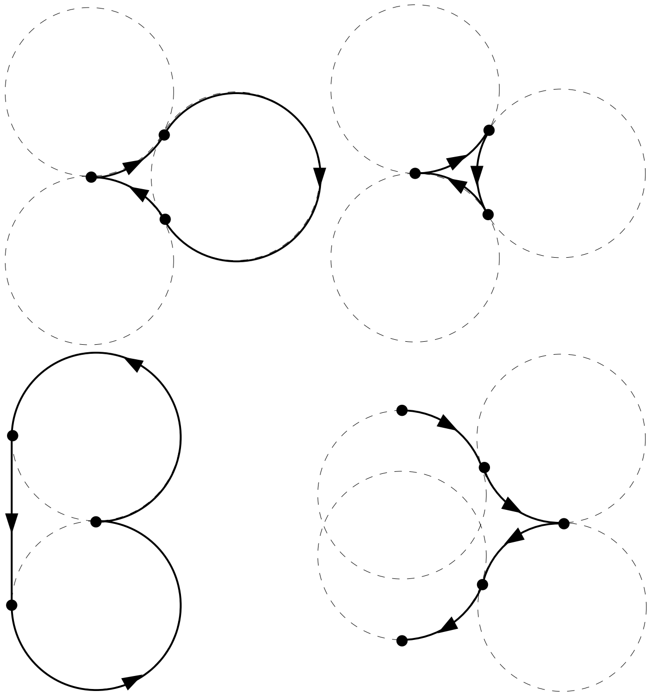


Figure 12: Some possible Reeds-Shepp curves for reversal in place and parallel parking.

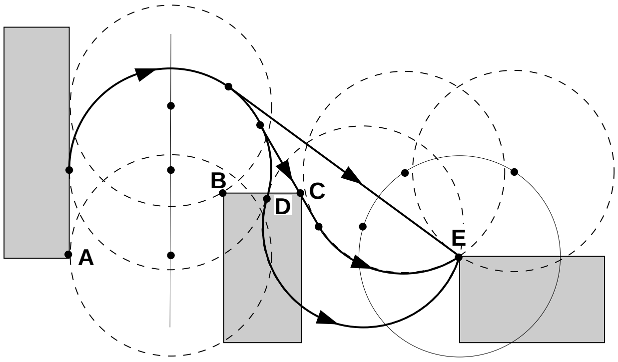


Figure 13: Events generating configuration space boundaries for a CSC Dubins' curve.

14 Optimal Planning for Point-like Cars

Jacobs and Canny [19, 20, 21] have adapted classical configuration space construction techniques to Dubins' curves to produce a pair of planners that find approximate minimum length paths for a point-like, limited-steering-lock car moving forward in the plane with open polygonal obstacles. The methods differ in the configuration space approximation method they use and in the robustness of the paths they produce.

The main results proved by Jacobs and Canny [19, 20, 21] are as follows:

1. If any path exists for the vehicle there is a minimum length path consisting of a finite number of smoothly joined Dubins' sections ending on obstacle edges or vertices, the start point or the goal.
2. There is a grid-based line sweep algorithm using Dijkstra search that finds a path within $\mathcal{O}(\delta)\%$ of the minimum length path in time $\mathcal{O}(\frac{n^3}{\delta}\log n + \frac{n^2}{\delta^2}\log\frac{n^2}{\delta})$.
3. There is a quadtree-based plane sweep algorithm that finds an '$\mathcal{O}(\delta)$-robust' path within $\mathcal{O}(\delta)\%$ of the minimum length δ-robust path in time $\mathcal{O}(n^4\log n + \frac{n^2}{\delta^2})$. Here, δ-robust means that perturbations of obstacles or endpoints of $\mathcal{O}(\delta)$ can not destroy the path.

Both planners are based on the construction of *configuration spaces* for the Dubins' curves. Since the segments of the optimal path always end on obstacle edges or vertices, each segment has only one degree of freedom at each end: along an edge the location can vary but the tangent direction is fixed, while at a vertex the orientation is free but the location is fixed. So for each pair of environment features (edges or vertices) and each class of Dubins' curve there is a two dimensional space of Dubins' segments between the features, whose 'points' are labelled by the location (along an edge) or orientation (at a vertex) of the start and end points. Some of these curves may intersect obstacles or fail to be realisable, but we can map out the 'configuration space obstacle' (CSO) corresponding to these forbidden regions of the segments' 'configuration space'.

Figure 13 illustrates some of these ideas. The configuration space for this CSC curve is swept out as the initial point slides along its edge and the final orientation pivots around its vertex. As this happens the initial and final curvature centres move along a line and an arc respectively, and the straight segment is tangent to the resulting circles. At certain points arcs or lines of the path cross vertices or edges of the environment (A,B,C) or arc or line endpoints collide with one another (D,E). These transition points represent potential CSO boundaries.

In a *tour de force* of arc-line geometry, Jacobs and Canny prove a series of theorems on the structure and construction of Dubins' configuration spaces. As an example, here is their 'four type theorem' for type-CSC curves [21]:

Figure 14: The configuration space of a CSC Dubins' curve constructed by the Jacobs-Canny algorithm (from [21]).

Figure 15: Some optimal paths for a forward moving point-like car produced by the Jacobs-Canny algorithm (from [21]).

Four types of constraint produce CSO boundaries for CSC curves:

1. An arc of the curve can hit an obstacle edge or vertex.
2. The straight section of the curve can hit an obstacle vertex.
3. An arc section can vanish (subtended angle $\rightarrow$ 0, jumping to 2π).
4. The straight section can vanish (length $\rightarrow$ 0).

If a given path is to be feasible, each of its segments must lie in the 'free space' region of the appropriate Dubins' configuration space. The search for an optimal path can now be formulated as a tree search over nodes corresponding to environment features labelled by the terminal location or orientation of the path on that feature. Every feature is connected to each of the others by each of the possible types of Dubins' curve and the search is pruned at curves that fail to travel through free 'windows' of configuration space.

A Dijkstra search of this graph is sufficient to find the minimal path, but to make this scheme concrete a suitable discretisation of the path spaces is required. Dividing the length of each edge and the orientation at each vertex into a fixed number of subdivisions produces a discretisation corresponding to a rectangular grid in the segments' configuration spaces.

The line sweep version of the planner constructs the quantized configuration space obstacle directly, by moving along each grid line classifying each point in turn. For speed it maintains a running tally of the CSO boundaries crossed along the line, so that each point (Dubins' segment) does not have to be tested against the raw geometry.

Figure 14 shows a CSC Dubins' configuration space constructed by the Jacobs-Canny line sweep algorithm. The horizontal and vertical boundaries represent collisions of the initial and final arcs with obstacles, some of the wave-like boundaries correspond to the vanishing of arc sections, and all the other boundaries arise from collisions of the straight section with obstacle vertices. Figure 15 shows some optimal paths generated by the method.

The quadtree version of the planner sweeps a line across the plane maintaining an ordered list of the positions of active CSO boundaries on the line. This allows it to generate a quadtree representing the completely free (i.e. neither occupied nor mixed) cells of the grid. The resulting half-cell-width of free space around each grid point (cell centre) is enough to derive δ-robustness.

Figure 16: The stages of planning a parallel parking manoeuvre in the Jacobs-Laumond scheme (from [26]).

15 The Jacobs-Laumond and Latombe Planners

The Jacobs-Canny planner produces optimal (minimum length) paths, but works only for a point-like car moving forwards among polygonal obstacles. Jacobs and Laumond [18, 27] have introduced a remarkable stochastic planner that uses an underlying *holonomic* planner, Reeds-Shepp curves, and randomized search to produce qualitatively good heuristic paths for a polygonal car moving forwards and backwards among polygonal obstacles. Latombe's modification of this [23, 10] is probably the best existing method for planning manoeuvres of realistic cars in realistic environments.

15.1 The Jacobs-Laumond Strategy

The Jacobs-Laumond and Latombe algorithms are examples of the following general strategy, which applies to any nonholonomic system for which local paths can be generated in the absence of obstacles (see fig. 16):

1. Choose any set of 'canonical' nonholonomic paths that connect arbitrary nearby system configurations without moving too far from either of them, and implement a collision checking routine for these paths. For a car the Reeds-Shepp curves are a good choice because they are guaranteed to be short and collision checking is easier for arc-line paths, but any similar family would do (e.g. spline-based paths or those generated in §9).
2. Ignore the nonholonomic constraints. Use any suitable planner to generate a path for a freely translating and rotating 'hovercraft' the shape of the vehicle. The path should clear all of the obstacles by some margin δ. If the planner produces obstacle-skimming paths, an obstacle growing or path relaxation stage is required to move the path away from the obstacles it touches.
3. Replace this infeasible holonomic path with a feasible nonholonomic one. If a canonical curve between the endpoints exists and is collision-free, replace the entire holonomic path segment with it. If not, recursively subdivide the segment and try again. This recursion is guaranteed to terminate because the endpoint separation eventually becomes much smaller than (any continuous function of) the clearance δ.
4. The resulting path is probably very jagged. Attempt to simplify it by repeatedly joining randomly generated pairs of intermediate configurations with collision-free canonical curves. For Reeds-Shepp curves this step shortens the path as well as simplifying it.

Although this strategy is clearly heuristic (i.e. the output is not 'optimal'), it is **complete** (guaranteed to find a path if one exists) whenever the underlying holonomic planner is guaranteed to find a path with non-zero clearance if one exists.

For a car the subdivision step produces a path of length $\mathcal{O}(l/\delta)$ with $\mathcal{O}(l/\delta^2)$ sections, where l is the length of the initial path. For example, to move sideways into a parallel park with forward clearance δ, a car may have to make repeated see-sawing manoeuvres of length $\mathcal{O}(\delta)$, each of which produces a sideways displacement of only $\mathcal{O}(\delta^2)$. More generally, it seems likely that for a (sufficiently regular) nonholonomic system of degree k (i.e. Lie brackets of up to k controls are required to span the state space directions) the resulting paths will have length $\mathcal{O}(l/\delta^{k-1})$ and con-

Figure 17: Some paths produced by Latombe's version of the Jacobs-Laumond scheme (from [23]).

tain $\mathcal{O}(l/\delta^k)$ manoeuvres [26]. Clearly, the initial planner should produce paths with as large a clearance as possible.

The original Jacobs-Laumond planner [18, 27] used a complete classical 'piano-movers' planner for the first step. As this produced obstacle-skimming paths, a heuristic perturbation method was developed to move the holonomic path away from the obstacles it touched. The resulting nonholonomic paths were quite usable, but sometimes more complex than necessary owing to the poor use of the available free space.

15.2 Latombe's Implementation

Latombe's version of the planner [23] uses the Barraquand-Latombe randomized planner [5, 24] for the first stage. This produces a heuristic potential field with relatively few local minima from a bitmap representation of the environment. It then follows the potential gradient in the usual way and uses an innovative randomized search technique to escape from any local minima it falls into. The potential is calculated by propagating wavefronts from the obstacle boundaries to produce an approximate free space Voronoi diagram, then propagating distances back from the goal along the Voronoi spines to produce the 'backbone' of the potential. To allow vehicle orientation to be controlled, potentials are calculated separately for several control points on the vehicle and combined heuristically.

The resulting 'holonomic' algorithm is probabilistically resolution complete and very fast in practice, and the paths it produces have almost maximal clearance since they roughly follow the Voronoi diagram. This simplifies the task of the subsequent stages of the scheme, so that the resulting planner can very quickly produce good paths, even in quite complex environments. Figure 17 shows some paths planned by Latombe's algorithm.

16 Summary

It is now possible to plan complex paths and manoeuvres for vehicles subject to nonholonomic constraints such as cars and trucks with arbitrary numbers of trailers. We have seen how to formulate the control laws and differential constraints for such systems in state space. Nonholonomic constraints are ones that fail to 'mesh' into integral surfaces, so that small local manoeuvres can 'undo' the constraint. The essence of this is captured by the Lie bracket, a kind of derivative operation acting on the control laws or constraints.

By iterating the Lie bracket one can find all of the displacements that can be produced by local manoeuvring. When *any* local displacement can be so generated, the nonholonomic system can follow arbitrary paths arbitrarily closely by energetic manoeuvring, so nonholonomic paths exist if and only if robust *unconstrained* paths exist. In this case, sequences of finite manoeuvres to drive the system between any two nearby configurations can be synthesized using several algebraic techniques.

In theory, optimal paths can be planned for general nonholonomic systems under the path-length or any other cost function, but this is can quickly become computationally intractable as the number of degrees of freedom is increased. For car-like vehicles the shortest paths are available in a simple closed form and this has resulted in several efficient, practical motion planning algorithms for cars under the shortest-path cost function.

17 Further Reading

Straightforward introductions to calculus on manifolds are available in a number of places. For the basic material on vector fields and differential forms a good starting point might be Arnold's excellent course on classical mechanics [3], followed by Abraham, Marsden and Ratiu [2]. For a more exhaustive treatment see Spivak [37]. Abraham and Marsden [1] is a comprehensive survey of calculus on manifolds, dynamical systems and nonlinear mechanics from a modern, geometric, synthetic point of view. (*N.B.* The first edition is more concrete and somewhat easier to read).

Macki and Strauss [30] is an accessible introduction to optimal control theory. Isidori [17] is a somewhat more advanced text on geometric control.

For a general introduction to motion planning read Latombe's monograph [24]. The collection by Li and Canny [29] is a good survey of current research in nonholonomic planning. In particular, the chapter by Laumond [26] provides many references and covers some aspects of nonholonomic planning more thoroughly than the current paper.

Acknowledgements

This paper grew out of a lecture presented during a study program on Computer Vision held at and supported by the Newton Institute of Mathematical Sciences, Cambridge, U.K. It leans heavily on material from the Li-Canny book [29] and has benefited by conversations with Roger Brockett. The author is supported by the European Community ESPRIT project SECOND.

A Brockett's Theorem

A system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ is **asymptotically stable** at $\boldsymbol{x}_0$ in $\boldsymbol{X}$ if the forward image of each neighbourhood $\mathcal{N}$ of $\boldsymbol{x}_0$ always stays inside $\mathcal{N}$ after some time $t_0(\mathcal{N})$ and finally converges to $\boldsymbol{x}_0$: $\mathrm{e}^{tf} \cdot \mathcal{N} \subseteq \mathcal{N}$ for all $t \geq t_0(\mathcal{N})$ and $\lim_{t\to\infty} \mathrm{e}^{tf} \cdot \mathcal{N} = \{\boldsymbol{x}_0\}$. Asymptotic stability is extremely important in the theory of general dynamical systems. The rough idea is that after youthful wanderings the system must settle down and tend to a limit in a well behaved way.

Brockett's theorem says the following [8]: Suppose that we want to find a continuously differentiable feedback law $\boldsymbol{u} \equiv \boldsymbol{u}(\boldsymbol{x})$ that 'parks' a continuously differentiable system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ at a state $\boldsymbol{x}_0$ with $\boldsymbol{f}(\boldsymbol{x}_0, 0) = 0$, and also makes $\boldsymbol{x}_0$ asymptotically stable so that small perturbations of the parking trajectory will be 'absorbed'. For such a control law to exist the following conditions must hold:

1. There must be a neighbourhood $\mathcal{N}$ of $\boldsymbol{x}_0$ within which each point $\boldsymbol{x}$ has *some* control sequence $\boldsymbol{u}_{\boldsymbol{x}}(t)$ steering it to $\boldsymbol{x}_0$ as $t \to \infty$.
2. The linearized system at $\boldsymbol{x}_0$ must have no uncontrollable unstable modes. There must be controls that can manipulate and stabilize any modes whose eigenvalues have positive real parts.
3. The mapping $\boldsymbol{f} : (\boldsymbol{x}, \boldsymbol{u}) \to \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ must be onto an open neighbourhood at 0. This means roughly that *all* the state space directions must be spanned by possible controlled motions near $\boldsymbol{x}_0$.

The third condition implies that *no* system with differential constraints can be stabilized to a point with a continuously differentiable feedback law, since the constrained controls can not span $\boldsymbol{T}_{\boldsymbol{x}_0}\boldsymbol{X}$. The proof of the theorem is topological and provides little insight into the obstruction to stability, but the general idea is that 'sideways' displacements (those not in $\mathrm{Span}(\frac{\mathrm{d}f}{\mathrm{d}u^i})$) have to be corrected by 'forwards' or 'backwards' manoeuvres (in the $\mathrm{Span}(\frac{\mathrm{d}f}{\mathrm{d}u^i})$ directions). But small shuffling manoeuvres require cusp-like reversals for which there is no continuously differentiable control law, while manoeuvres without reversals must 'go around in a loop and come back for another try' and can not be interpolated smoothly to zero as the start point is moved towards the fixed point $\boldsymbol{x}_0$.

B The Maximum Principle

Pontryagin's Maximum Principle for time independent dynamics can be briefly stated as follows [35]:

1. Given dynamics $\dot{x^a} = f^a(\boldsymbol{x}, \boldsymbol{u})$ and initial and final states $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, designate the cost function by $f^0(\boldsymbol{x}, \boldsymbol{u})$ and prepend a component x^0 to $\boldsymbol{x}$ to contain the running cost total: $\dot{x}^0 = f^0(\boldsymbol{x}, \boldsymbol{u}) \equiv c(\boldsymbol{x}, \boldsymbol{u})$ so that $x^0(t) = \int^t c(\boldsymbol{x}, \boldsymbol{u})dt$.
2. Introduce 'momentum' variables p_a dual to the extended $\boldsymbol{x}$ vector and define a Hamiltonian $\boldsymbol{H}(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{u}) \equiv \sum_a p_a f^a(\boldsymbol{x}, \boldsymbol{u})$. For any given $\boldsymbol{u}(t)$

and initial conditions $\boldsymbol{p}(t_0), \boldsymbol{x}(t_0)$ we can integrate the Hamiltonian equations of motion:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \frac{\partial \boldsymbol{H}(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{p}} \\ \dot{\boldsymbol{p}} &= -\frac{\partial \boldsymbol{H}(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}} \end{aligned}$$

The first equation gives the system dynamics and is independent of $\boldsymbol{p}$. The second tells how the momentum changes with time and can be integrated once $\boldsymbol{x}(t)$ is known. Note that p_0 is conserved (constant with time) because the Hamiltonian is independent of the cost x^0, and the Hamiltonian dynamics itself guarantees that the 'energy' $\boldsymbol{H}(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{u})$ is conserved.

3. Now start from $\boldsymbol{x}_0$ and an arbitrary initial momentum $\boldsymbol{p}(t_0)$, and at each time step *apply the control which maximizes* $\boldsymbol{H}$: $\boldsymbol{u} \equiv \arg\max_{\boldsymbol{u} \in \boldsymbol{U}} \boldsymbol{H}(\boldsymbol{p}(t), \boldsymbol{x}(t), \boldsymbol{u})$. Update $\boldsymbol{x}$ and $\boldsymbol{p}$ with Hamilton's equations as usual, and continue.

4. Eventually, the path will miss $\boldsymbol{x}_1$. Adjust $\boldsymbol{p}(t_0)$ and repeat the process until the path passes through $\boldsymbol{x}_1$. Since $\boldsymbol{H}$ is homogeneous in $\boldsymbol{p}$ it is sufficient to fix $p_0(t_0) = -1$. Do not bother to consider values of $\boldsymbol{p}(t_0)$ for which $\max_{\boldsymbol{u}} \boldsymbol{H}(\boldsymbol{p}(t_0), \boldsymbol{x}_0, \boldsymbol{u})$ is nonzero as they can not be optimal.

5. If the previous step succeeds, Pontryagin's main result [35] is that the resulting path is locally optimal and *may* be globally optimal (i.e., the above conditions are necessary but not sufficient for optimality). In practice the chances are high that the best candidate trajectory passing through $\boldsymbol{x}_1$ is in fact the optimal one, but in general this is a delicate mathematical question.

6. There are usually certain critical values of $\boldsymbol{x}$ and $\boldsymbol{p}$ at which the maximising control suddenly jumps from one point of $\boldsymbol{U}$ to another. For bounded controls these points tend to be on the boundaries of $\boldsymbol{U}$. So the optimal control schedule typically consists of a piecewise continuous sequence of hard-limited controls, with abrupt jumps between constraint boundaries at critical times: 'bang-bang' controls. The momenta p_a are called **switching functions**. The key information encoded in the implicit initial condition $\boldsymbol{p}(t_0)$ is the switching schedule for the solution.

References

[1] Ralph Abraham and Jerrold E. Marsden. *Foundations of Mechanics*. Benjamin-Cummings, 2^{nd} edition, 1978. (1^{st} edition, 1967).

[2] Ralph Abraham, Jerrold E. Marsden, and Tudor Ratiu. *Manifolds, Tensor Analysis and Applications*. Global Analysis Pure and Applied series. Addison-Wesley, 1983.

[3] V. I. Arnold. *Mathematical Methods of Classical Mechanics*. Number 60 in Graduate Texts in Mathematics. Springer-Verlag, 1978.

[4] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. In *IEEE International Conference on Robotics and Automation*, pages 2328–35, Sacramento, CA, April 1991.

[5] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–49, 1991.

[6] J. D. Boissonnat, A. Cerezo, and J. Leblong. Shortest paths of bounded curvature in the plane. Technical report, INRIA, Sophia Antipolis, July 1991.

[7] N. Bourbaki. *Groupes et Algèbres de Lie*, chapter 2–3. CCLS Diffusion, Paris, 1972.

[8] R. W. Brockett. Asymptotic stability and feedback stabilization. In R. W. Brockett, R. Millman, and H. Sussmann, editors, *Differential Geometry and Geometrical Control Theory*, volume 27 of *Progress in Mathematics*, pages 181–91, Boston, 1983. Birkhäuser.

[9] J. Canny, B. Donald, J. Reif, and P. Xavier. On the complexity of kinodynamic planning. In *29th Symposium on the Foundations of Computer Science*, White Plains, NY, 1988.

[10] W. Choi and J.-C. Latombe. A reactive architecture for planning and executing robot motions with incomplete knowledge. In *IEEE International Workshop on Intelligent Robot Systems (IROS)*, Osaka, November 1991.

[11] W. L. Chow. Uber systeme von lenearen partiellen differentialliechungen ester ordung. *Math. Ann.*, 117:98–105, 1939.

[12] J.-M. Coron. Global asymptotic stabilization for controllable systems without drift. Technical report, 1991.

[13] B. Donald and P. Xavier. A provably good approximation algorithm for optimal-time trajectory planning. In *IEEE International Conference on Robotics and Automation*, pages 958–63, 1989.

[14] 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*, 79:497–516, 1957.

[15] C. Fernandes, L. Gurvits, and Z. X. Li. Optimal nonholonomic motion planning for a falling cat. In Li and Canny [29], pages 379–421.

[16] S. Fleury, P. Souères, J.-P. Laumond, and R. Chatila. Primitives for smoothing mobile robot trajectories. In *IEEE International Conference on Robotics and Automation*, pages 832–9, 1993.

[17] A. Isidori. *Nonlinear Control Sytems*. Communication and Control Engineering. Springer-Verlag, 1989.

[18] P. Jacobs, J.-P. Laumond, and M. Taïx. Efficient motion planning for nonholonomic mobile robots. In *IEEE International Workshop on Intelligent Robot Systems (IROS)*, Osaka, November 1991.

[19] Paul Jacobs and John Canny. Planning smooth paths for mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 2–7, 1989.

[20] Paul Jacobs and John Canny. Robust motion planning for mobile robots. In *IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, May 1990.

[21] Paul Jacobs and John Canny. Planning smooth paths for mobile robots. In Li and Canny [29], pages 271–342.

[22] G. Lafferriere and H. J. Sussmann. A differential geometric approach to motion planning. In Li and Canny [29], pages 235–70.

[23] J.-C. Latombe. A fast path planner for a car-like indoor mobile robot. In *National Conference on Artificial Intelligence*, Anaheim, California, July 1991. AAAI.

[24] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[25] J.-P. Laumond. Controllability of a multibody mobile robot. In *International Conference on Advanced Robotics*, Pisa, June 1991.

[26] J.-P. Laumond. Singularities and topological aspects in nonholonomic motion planning. In Li and Canny [29], pages 149–99.

[27] J.-P. Laumond, M. Taïx, and P. Jacobs. A motion planner for car-like robots based on a mixed global/local approach. In *IEEE International Workshop on Intelligent Robot Systems (IROS)*, pages 765–73, 1990.

[28] J. D. Lawrence. *A Catalog of Special Plane Curves*. Dover, 1972.

[29] Zexiang Li and J. F. Canny, editors. *Nonholonomic Motion Planning*. Kluwer Academic Publishers, 1993.

[30] J. Macki and A. Strauss. *An Introduction to Optimal Control Theory*. Undergraduate Texts in Mathematics series. Springer-Verlag, 1982.

[31] R. M. Murray and S. S. Sastry. Grasping and manipulation using multifingered robot hands. In R. W. Brockett, editor, *Robotics*, volume 41 of *Proceedings of Symposia in Applied Mathematics*, pages 91–128. American Mathematical Society, 1990.

[32] R. M. Murray and S. S. Sastry. Steering nonholonomic systems using sinusoids. In *IEEE Control and Decision Conference*, 1990.

[33] R. M. Murray and S. S. Sastry. Steering nonholonomic control systems using sinusoids. In Li and Canny [29], pages 23–51.

[34] E. G. Papadopoulos. Nonholonomic behaviour in free-floating space manipulatiors and its utilisation. In Li and Canny [29], pages 423–45.

[35] L. S. Pontryagin. *The Mathematical Theory of Optimal Processes*, volume 55 of *International Series of Monographs in Pure and Applied Mathematics*. Pergamon, Oxford, 1986.

[36] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145:367–393, 1990.

[37] M. Spivak. *A Comprehensive Treatise on Differential Geometry*. Publish or Perish, Boston, Mass., 1974.

[38] H. J. Sussmann. Lie brackets, real analyticity and geometric control. In R. W. Brockett, R. Millman, and H. Sussmann, editors, *Differential Geometry and Geometrical Control Theory*, volume 27 of *Progress in Mathematics*, pages 1–116, Boston, 1983. Birkhäuser.

[39] 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. Technical Report SYCON-91-10, Rutgers University, 1991.