Multimedia Contents

47. Motion Planning and Obstacle Avoidance

Javier Minguez, Florant Lamiraux, Jean-Paul Laumond

This chapter describes motion planning and obstacle avoidance for mobile robots. We will see how the two areas do not share the same modeling background. From the very beginning of motion planning, research has been dominated by computer sciences. Researchers aim at devising well-grounded algorithms with well-understood completeness and exactness properties.

The challenge of this chapter is to present both nonholonomic motion planning (Sects. 47.1–47.6) and obstacle avoidance (Sects. 47.7–47.10) issues. Section 47.11 reviews recent successful approaches that tend to embrace the whole problem of motion planning and motion control. These approaches benefit from both nonholonomic motion planning and obstacle avoidance methods.

The introduction of nonholonomic constraints has entailed revisiting these algorithms via the introduction of differential geometry materials. Such a combination has been made possible for certain classes of systems, the so-called small-time controllable ones. It remains that the underlying hypothesis of motion planning algorithms is the knowledge of a global and accurate map of the environment. More than that the considered system is a formal system of equations that does not account for the entire physical system: uncertainties in the world or system modeling are not considered. Such hypotheses are too strong in practice. This is why other complementary research has been done in parallel in a more pragmatic but realistic manner. Such research deals with obstacle avoidance. The problem here is not to deal with complicated systems like a car with multiple trailers. The considered systems are much simpler with respect to their geometric shape. The problem considers sensor-based motions to face the physical issues of a real system navigating in a real world better than motion planning algorithms. How can we *navigate* toward a goal in a cluttered environment when the obstacles to avoid have just been discovered in real time? This is the question that obstacle avoidance addresses.

The appearance of mobile robots in the late 1960s early 1970s initiated a new research domain: autonomous navigation. It is interesting to note that the first navigation systems were published at the very first International Joint Conferences on Artificial Intelligence (IJCAI 1969). These systems were based on seminal ideas, which have been very fruitful in the development of robot motion planning algorithms. For instance, in 1969, the mobile robot Shakey used a grid-based approach to model and explore the environment [47.1], in 1977 Jason used a visibility graph built from the corners of the obstacles [47.2], and in 1979 Hilare decomposed the environment into collision-free convex cells [47.3].

In the late 1970s, studies of robot manipulators popularized the notion of the configuration space of a mechanical system [47.4]. In the configuration space the *piano* becomes a point. The motion planning problem for a mechanical system was thus reduced to finding a path for a point in the configuration space. The way was open to extending the seminal ideas and to developing new and well-grounded algorithms (see *Latombe*'s book [47.5]).

One decade later, the notion of nonholonomic systems (also borrowed from mechanics) appeared in the literature [47.6] on robot motion planning through the problem of car parking. This problem had not been solved by the pioneering works on mobile robot navigation. Nonholonomic motion planning then became an attractive research field [47.7].

Besides this research effort in path planning, work was initiated in order to make robots move out of their initially artificial environments where the world was cylindrical and composed of wooden vertical boards. Robots started to move in laboratory buildings, with people walking around. Inaccurate localization, uncertain and incomplete maps of the world, and unexpected moving or static obstacles made roboticists aware of the gap between planning a path and executing a motion. Since then the domain of obstacle avoidance has been very active.

In the 2000s a lot of effort has been made toward the integration of motion planning and obstacle avoidance. This effort was stimulated by the DARPA Urban challenge competition. The most successful integration was done by the Carnegie Mellon University team, who won the competition [47.8].

47.1 Nonholonomic Mobile Robots: Where Motion Planning Meets Control Theory

Nonholonomic constraints are nonintegrable linear constraints over the velocity space of a system. For instance, the rolling without slipping constraint of a differentially-driven mobile robot (Fig. 47.1) is linear with respect to the velocity vector (vector of linear and angular velocities) of the differentially-driven robot and is non integrable (it cannot be integrated into a constraint over the configuration variables). As a consequence, a differentially-driven mobile robot can go anywhere but not following any trajectory. Other types of nonholonomic constraints arise when considering second-order differential equations such as the conservation of inertial momentum. A number of papers investigate the famous falling cat problem or free-floating robots in space [47.7]. This chapter is devoted to nonholonomic constraints for mobile robots with wheels.

While the constraints due to obstacles are expressed directly in the configuration space, that is a manifold, nonholonomic constraints are expressed in the tangent space. In the presence of a linear kinematic constraint, the first question that naturally arises is: *does this constraint reduce the space* reachable *by the system?* This question can be answered by studying the structure of the distribution spanned by the Lie algebra of the control system.

Even in the absence of obstacles, planning admissible motions (i. e., that satisfy the kinematic constraints) for a nonholonomic system between two configurations is not an easy task. Exact solutions have been proposed only for some classes of systems but a lot of systems remain without exact solution. In the general case, however, approximate solutions can be used.

The motion planning problem for a nonholonomic system can be stated as follows: given a map of the environment with obstacles in the workspace, a robot subject to nonholonomic constraints, an initial configuration and a goal configuration, find an admissible collision-free path between the initial and goal configurations. Solving this problem requires taking into account both the configuration space constraints due to obstacles and the nonholonomic constraints. The tools developed to address this issue thus combine motion planning and control theory techniques. Such a combination is possible for the class of so-called small-time controllable systems due to topological arguments (Theorem 47.2 in Sect. 47.3).

47.2 Kinematic Constraints and Controllability

In this section, we give the main definition of controllability, using *Sussman*'s terminology [47.9].

47.2.1 Definitions

Let us denote by CS the configuration space of dimension n of a given mobile robot and by $\boldsymbol{q}$ the configuration of this robot. If the robot is mounted on wheels, it is subject to kinematic constraints, linear in the velocity vector

$$\omega_i(\boldsymbol{q})\dot{\boldsymbol{q}} = 0\,, \quad i \in \{1,\dots,k\}\,.$$

We assume that these constraints are linearly independent for any $\boldsymbol{q}$. Equivalently, for each $\boldsymbol{q}$, there exists $m = n-k$ linearly independent vectors $f_1(\boldsymbol{q}),\dots,f_m(\boldsymbol{q})$ such that the above constraints are equivalent to

$$\exists(u_1,\dots,u_m) \in \mathbb{R}^m\,, \quad \dot{\boldsymbol{q}} = \sum_{i=1}^{m} u_i f_i(\boldsymbol{q})\,. \tag{47.1}$$

Let us note that the choice of vectors $f_i(\boldsymbol{q})$ is not unique. Fortunately, all the following developments are valid whatever choice we make. Moreover, if the linear constraints are smooth, vector fields $f_1,\dots,f_m$ can be chosen smooth with respect to $\boldsymbol{q}$. We assume this condition from now on.

Let us define by $\mathcal{U}$ a compact subset of $\mathbb{R}^m$. We denote by Σ the control system defined by (47.1), with $(u_1,\dots,u_m) \in \mathcal{U}$.

Definition 47.1
A local and small-time controllability:

1. Σ is *locally controllable about configuration* $\boldsymbol{q}$ iff the set of configurations reachable from $\boldsymbol{q}$ by an admissible trajectory contains a neighborhood of $\boldsymbol{q}$.
2. Σ is *small-time controllable about configuration* $\boldsymbol{q}$ iff the set of configurations reachable from $\boldsymbol{q}$ by an admissible trajectory in time less than T contains a neighborhood of $\boldsymbol{q}$ for any T.

$f_1,\dots,f_m$ are called *control vector fields* of Σ. A system's small-time controllable about each configuration is said to be *small-time controllable*.

47.2.2 Controllability

Checking the controllability properties of a system requires the analysis of the control Lie algebra associated with the system. Let us illustrate in an informal way what the Lie bracket of two vector fields is. Consider two basic motions: *go along a straight line* and *turn on the spot*, supported by vector fields denoted by f and g, respectively. Now consider the following combination: go forward during a time t, turn clockwise during the same time t, go backward during the same time t and then turn counterclockwise during the time t. The system reaches a configuration that is not the starting one. Of course when t tends to 0 the goal configuration is very close to the starting one. The direction indicated by such goal configurations when t tends to 0 correspond to a new vector field, which is the Lie bracket of f and g. In a more mathematical formulation, the Lie bracket $[f,g]$ of two vector fields f and g is defined as being the vector field $\partial f.g - \partial g.f$. The k-th coordinate of $[f,g]$ is

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

The following theorem [47.10] gives a powerful result for symmetric systems (a system is said to be symmetric when $\mathcal{U}$ is symmetric with respect to the origin).

Theorem 47.1
A symmetric system is small-time controllable about configuration $\boldsymbol{q}$ iff the rank of the vector space spanned by the family of vector fields f_i together with all their brackets is n at $\boldsymbol{q}$.

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 is proposed in [47.11, 12].

47.2.3 Example: The Differentially-Driven Mobile Robot

To illustrate the notions developed in this section, we consider the differentially-driven mobile robot displayed in Fig. 47.1. The configuration space of this robot is $\mathbb{R}^2 \times S^1$, and a configuration can be represented by $\boldsymbol{q} = (x, y, \theta)$ where (x, y) is the position in the horizontal plane of the center of the wheel axis of the robot and θ the orientation with respect to the x-axis. The rolling without slipping kinematic constraint

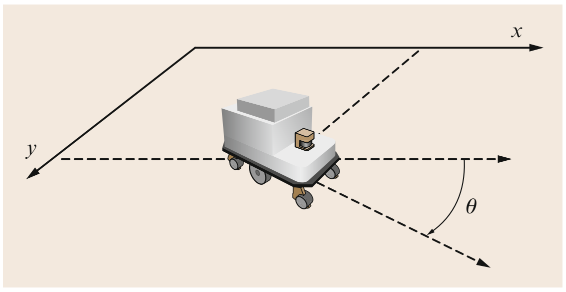

Fig. 47.1 A differentially-driven mobile robot is subject to one linear kinematic constraint, due to the rolling without slipping constraint of the wheel axis

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

is linear with respect to the velocity vector $(\dot{x}, \dot{y}, \dot{\theta})$. Therefore, the subspace of admissible velocities is spanned by two vector fields, for instance,

$$f_1(\boldsymbol{q}) = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} \quad \text{and} \quad f_2(\boldsymbol{q}) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} . \tag{47.2}$$

The Lie bracket of these two vector fields is

$$f_3(\boldsymbol{q}) = \begin{pmatrix} \sin\theta \\ -\cos\theta \\ 0 \end{pmatrix} .$$

This implies that about any configuration $\boldsymbol{q}$, the rank of the vector space spanned by $f_1(\boldsymbol{q}), f_2(\boldsymbol{q}), f_3(\boldsymbol{q})$ is 3 and, therefore, that the differentially-driven mobile robot is small-time controllable.

47.3 Motion Planning and Small-Time Controllability

Motion planning raises two problems: the first one addresses the *existence* of a collision-free admissible path; this is the decision problem. The second one addresses the *computation* of such a path; this is the complete problem.

47.3.1 The Decision Problem

From now on, we assume that the set of collision-free configurations is an open subset. This implies that contact configurations are assumed in collision.

Theorem 47.2
The existence of a collision-free admissible path for a symmetric small-time controllable mobile robot between two configurations is equivalent to the existence of a collision-free (not necessarily admissible) path between these configurations.

Proof: Let us consider a not necessarily admissible collision-free path between two configurations $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ as a continuous mapping Γ from interval $[0,1]$ into the configuration space CS such that:

1. $\Gamma(0) = \boldsymbol{q}_1$, $\Gamma(1) = \boldsymbol{q}_2$,
2. for any $t \in [0,1]$, $\Gamma(t)$ is collision free.

Point 2 implies that for any t there exists a neighborhood $U(t)$ of $\Gamma(t)$ included in the collision-free subset of the configuration space.

Let us denote by $\varepsilon(t)$ the bigger lower bound of the time to collision of all the trajectories starting from

$\Gamma(t)$. As the control vector $(u_1, \ldots, u_m)$ remains in the compact set $\mathcal{U}$, $\varepsilon(t) > 0$.

As the system is small-time controllable about $\Gamma(t)$, the set reachable from $\Gamma(t)$ in time less than $\varepsilon(t)$ is a neighborhood of $\Gamma(t)$, which we denote by $V(t)$.

The collection $\{V(t), t \in [0,1]\}$ is an open covering of the compact set $\{\Gamma(t), t \in [0,1]\}$. Therefore, we can extract a finite covering, $\{V(t_1), \ldots, V(t_l)\}$, where $t_1 = 0 < t_2 < \ldots < t_{l-1} < t_l = 1$ such that for any i between 1 and $l-1$, $V(t_i) \cap V(t_{i+1}) \neq \emptyset$. For each i between 1 and $l-1$, we choose one configuration $\boldsymbol{r}_i$ in $V(t_i) \cap V(t_{i+1})$. As the system is symmetric, there exists an admissible collision-free path between $\boldsymbol{q}(t_i)$ and $\boldsymbol{r}_i$ and between $\boldsymbol{r}_i$ and $\boldsymbol{q}(t_{i+1})$. The concatenation of these paths is a collision-free admissible path between $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$. ■

47.3.2 The Complete Problem

In the former section, we have established that the decision problem, i.e., determining whether there exists a collision-free admissible path between two configurations, is equivalent to determining whether the configurations lie in the same connected component of the collision-free configuration space. In this section we present the tools necessary to solve the complete problem. These tools blend ideas from the classical motion planning problem addressed in Chap. 7 and from open loop control theory, but require specific developments that we are going to present in the next section. Two main approaches have been devised in order to plan admissible collision-free motions for nonholonomic systems. The first one proposed by [47.13] exploits the idea of the proof of Theorem 47.2 by recursively approximating a not necessarily admissible collision-free path by a sequence of feasible paths. The second approach replaces the local method of probabilistic roadmap method (PRM) algorithms (Chap. 7) by a local steering method that connects configuration pairs by admissible paths (VIDEO 707).

Both approaches use a steering method. Before briefly describing them, we give the definition of a local steering method.

Definition 47.2
A local steering method for system Σ is a mapping

$$S_{\text{loc}} : CS \times CS \to C^1_{\text{pw}}([0,1], CS)$$
$$(\boldsymbol{q}_1, \boldsymbol{q}_2) \mapsto S_{\text{loc}}(\boldsymbol{q}_1, \boldsymbol{q}_2)$$

5 where $S_{\text{loc}}(\boldsymbol{q}_1, \boldsymbol{q}_2)$ is a piecewise continuously differentiable curve in CS satisfying the following properties:

1. $S_{\text{loc}}(\boldsymbol{q}_1, \boldsymbol{q}_2)$ satisfies the kinematic constraints associated to Σ,
2. $S_{\text{loc}}(\boldsymbol{q}_1, \boldsymbol{q}_2)$ connects $\boldsymbol{q}_1$ to $\boldsymbol{q}_2$: $S_{\text{loc}}(\boldsymbol{q}_1, \boldsymbol{q}_2)(0) = \boldsymbol{q}_1$, $S_{\text{loc}}(\boldsymbol{q}_1, \boldsymbol{q}_2)(1) = \boldsymbol{q}_2$.

Approximation of a Not Necessarily Admissible Path

A not necessarily admissible collision-free path $\Gamma(t), t \in [0,1]$ connecting two configurations and a local steering method S_{loc} being given, the approximation algorithm proceeds recursively, by calling function `approximation` defined by Algorithm 47.1 with input Γ, 0 and 1.

Algorithm 47.1
`approximation` function: inputs are a path Γ and two abscissas $t1$ and $t2$ along this path

```
if $S_{loc}(\Gamma(t1), \Gamma(t2))$ collision free then
    return $S_{loc}(\Gamma(t1), \Gamma(t2))$
else
    return concat(approximation($\Gamma$,t1,(t1+t2)/2),
                  approximation($\Gamma$,(t1+t2)/2,t2))
end if
```

Sampling-Based Roadmap Methods

Most sampling-based roadmap methods as described in Chap. 6 can be adapted to nonholonomic systems by replacing the connection method between pairs of configurations by a local steering method. This strategy is rather efficient for PRM algorithms. For the rapidly-exploring random tree (RRT) method, the efficiency strongly depends on the metric used to choose the nearest neighbor. The distance function between two configurations needs to account for the length of the path returned by the local steering method to connect these configurations [47.14].

47.4 Local Steering Methods and Small-Time Controllability

The approximation algorithm described in the former section is recursive and raises the completeness question: does the algorithm finish in finite time or may it fail to find a solution?

A sufficient condition for the approximation algorithm to find a solution in a finite number of iterations is that the local steering method accounts for small-time controllability.

Definition 47.3

A local steering method S_{loc} accounts for the small-time controllability of system Σ iff

For any $\boldsymbol{q} \in CS$, for any neighborhood U of $\boldsymbol{q}$, there exists a neighborhood V of $\boldsymbol{q}$ such that for any $\boldsymbol{r} \in V$, $S_{\text{loc}}(\boldsymbol{q}, \boldsymbol{r})([0,1]) \subset U$.

In other words, a local steering method accounts for the small-time controllability of a system if it produces paths getting closer to the configurations it connects when these configurations get closer to each other.

This property is also sufficient for probabilistic completeness of roadmap sampling-based methods.

47.4.1 Local Steering Methods Accounting for Small-Time Controllability

Constructing a local steering method that accounts for small-time controllability is a difficult task that has been achieved only for a few classes of systems. Most mobile robots studied in the domain of motion planning are wheeled mobile robots towing trailers or not.

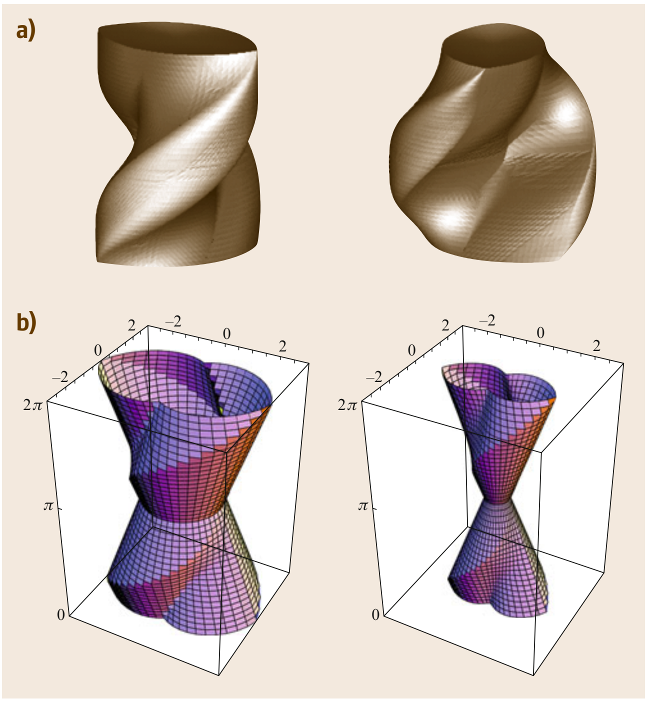

Fig. 47.2 (a) Two perspective views of an RS (Reeds and Shepp) ball: set of configurations reachable by a path of length less than a given distance for the Reeds and Shepp car. **(b)** Perspective views of two DD (differentially driven) balls: set of configurations reachable by a path of a length less than a given distance for the differentially-driven robot. Orientation θ is represented on z-axis

Steering Using Optimal Control

The simplest system, namely the differentially-driven robot presented in Sect. 47.2.3 with bounded velocities or the so-called *Reeds* and *Shepp* [47.15] car with bounded curvature have the same control vector fields (47.2). The difference lies in the domain of the control variables:

- $-1 \leq u_1 \leq 1$, $|u_2| \leq |u_1|$ for RS car,
- $|u_1| + b|u_2| \leq 1$ for the differentially-driven robot,

where b is half of the distance between the right and left wheels. For these systems, a local steering method accounting for small-time controllability can be constructed using optimal control theory. For any admissible path defined over an interval I of one of these systems, we define a length as follows

$$\int_I |u_1| \quad \text{for the RS car}\,,$$

$$\int_I |u_1| + b|u_2| \quad \text{for the differentially-driven robot}\,.$$

The length of the shortest path between two configurations in both cases defines a metric over the configuration space. The synthesis of the shortest paths, i. e., the determination of the shortest path between any pair of configurations was achieved by [47.16] for the RS car and later by [47.17] for the differentially-driven robot. Figure 47.2 shows a representation of the balls corresponding to these metrics.

Optimal control naturally defines a local steering method that associates a shortest path between these configurations to any pair of configurations. Let us note that the shortest path is unique between most pairs of configurations. A general result states that the collection of balls of radius $r > 0$ centered about $\boldsymbol{q}$ induced by nonholonomic metrics constitutes an increasing collection of neighborhoods of $\boldsymbol{q}$, the intersection of which is $\{\boldsymbol{q}\}$. This property directly implies that local steering methods based on shortest paths account for small-time controllability.

The main advantage of optimal control is that it provides both a local steering method and a distance metric consistent with the steering method. This makes the steering methods well suited for path planning algorithms designed for holonomic systems and using a distance function, such as RRT (Chap. 7), for instance.

Unfortunately, the synthesis of the shortest paths has been realized only for the two simple systems described in this section. For more complex systems, the problem remains open.

The main drawback of the shortest path-based steering methods described in this section is that input functions are not continuous. This requires an additional step to compute a time-parameterization of the paths before motion execution. Along this time-parameterization, input discontinuities force the robot to stop. For instance, to follow two successive arcs of circles of opposite curvature a mobile robot needs to stop between the arcs of circle in order to ensure continuity of the linear and angular velocities u_1 and u_2.

Steering Chained-Form Systems

Some classes of systems can be put into the form called the *chained-form* system by a change of variable

$$\dot{z}_1 = u_1\,, \tag{47.3}$$
$$\dot{z}_2 = u_2\,, \tag{47.4}$$
$$\dot{z}_3 = z_2 u_1\,, \tag{47.5}$$
$$\vdots \quad \vdots$$
$$\dot{z}_n = z_{n-1} u_1\,. \tag{47.6}$$

Let us consider the following inputs [47.18]

$$\begin{cases} u_1(t) &= a_0 + a_1 \sin\omega t\,, \\ u_2(t) &= b_0 + b_1 \cos\omega t \\ & \quad + \ldots + b_{n-2}\cos(n-2)\omega t\,. \end{cases} \tag{47.7}$$

Let $Z^{\text{start}} \in \mathbb{R}^n$ be a starting configuration. Each $z_i(1)$ can be computed from the coordinates of Z^{start} and parameters $(a_0, a_1, b_0, b_1, \ldots, b_{n-2})$. For a given $a_1 \neq 0$ and a given configuration Z^{start}, the mapping from $(a_0, b_0, b_1, b_2, b_3)$ to $Z(1)$ is a C^1-diffeomorphism at the origin; the system is then invertible. For n smaller or equal to 5, parameters $(a_0, b_0, b_1, \ldots, b_{n-2})$ can be analytically computed from the coordinates of the two configurations Z^{start} and Z^{goal}. The corresponding sinusoidal inputs steer the system from Z^{start} to Z^{goal}. The shape of the path only depends on parameter a_1. Each value of a_1 thus defines a local steering method denoted by $S_{\text{sin}}^{a_1}$. None of these steering methods account for small-time controllability since for any $Z \in \mathbb{R}^n$, $S_{\text{sin}}^{a_1}(Z,Z)([0,1])$ is not reduced to $\{Z\}$. To construct a local steering method accounting for small-time controllability from the collection of $S_{\text{sin}}^{a_1}$, we need to make a_1 depend on the configurations Z_1 and Z_2 that we want to connect

$$\lim_{Z^2 \to Z^1} a_1(Z^1, Z^2) = 0\,,$$
$$\lim_{Z^2 \to Z^1} a_0\left[Z^1, Z^2, a_1(Z^1, Z^2)\right] = 0\,,$$
$$\lim_{Z^2 \to Z^1} b_i\left[Z^1, Z^2, a_1(Z^1, Z^2)\right] = 0\,.$$

Such a construction is achieved in [47.19].

Steering Feedback-Linearizable Systems

The concept of *feedback linearizability* (or differential flatness) was introduced by *Fliess* et al. [47.20, 21].

A system is said to be *feedback linearizable* if there exists an output (i. e., function of the state, input and input derivatives) called the *linearizing output*, such that the state and the input of the system is a function of the linearizing output and its derivatives. The dimension of the linearizing output is the same as the dimension of the input.

Let us illustrate this notion with a simple example. We consider a differentially-driven mobile robot towing a trailer hitched on top of the wheel axis of the robot, as displayed in Fig. 47.3. The tangent to the curve followed by the center of the wheel axis of the trailer gives the orientation of the trailer. From the orientation of the trailer along the curve, we can deduce the curve followed by the center of the robot. The tangent to the curve followed by the center of the robot gives the orientation of the robot. Thus the linearizing output of this system is the center of the wheel axis of the trailer. By differentiating the linearizing output twice, we can reconstruct the configuration of the system.

Feedback linearizability is very interesting for steering purposes. Indeed, the linearizing output is not subject to any kinematic constraints. Therefore, if we know the relation between the state and the linearizing output, planning an admissible path between two configurations simply consists in building a curve in $\mathbb{R}^m$,

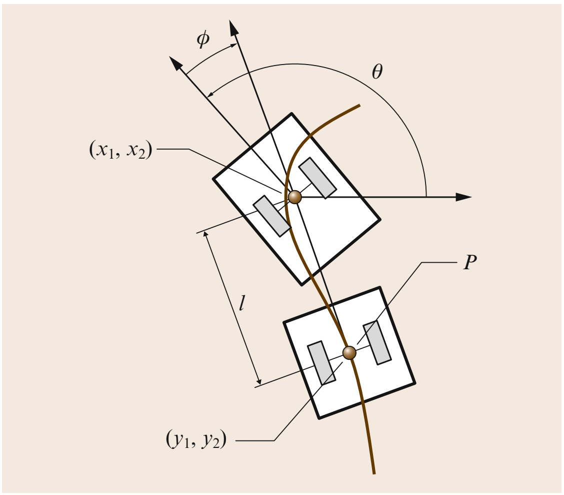

Fig. 47.3 A differentially-driven robot towing a trailer. Hitched on top of the wheel axis of the robot is a feedback-linearizable system. The linearizing output is the center of the wheel axis of the trailer. The configuration of the system can be reconstructed by differentiating the curve $\boldsymbol{y}(s)$, where s is an arc-length parameterization, followed by the linearizing output. The orientation of the trailer is given by $\tau = \arctan(\dot{y}_2/\dot{y}_1)$. The angle between the robot and the trailer is given by $\phi = -l \arctan(\mathrm{d}\tau/\mathrm{d}s)$. l is the length of the trailer connection

where m is the dimension of the input with differential constraints at both ends. This problem can be easily solved using, for instance, polynomials.

For two input driftless systems like Σ, the linearizing output only depends on state $\boldsymbol{q}$. The state $\boldsymbol{q}$ depends on the linearizing output through the parameterization invariant values, namely, the linearizing output $\boldsymbol{y}$, the orientation τ of the vector tangent to the curve followed by $\boldsymbol{y}$ and the successive derivatives of τ with respect to the curvilinear abscissa s.

Thus, the configuration of a two input feedback-linearizable driftless system of dimension n can be represented by a vector $(\boldsymbol{y}, \tau, \tau^1, \ldots, \tau^{n-3})$ representing the geometric properties of the curve followed by the linearizing output along an admissible path passing through the configuration.

Therefore, designing a local steering method for such a system is equivalent to associating to any pair of vectors $(\boldsymbol{y}_1, \tau_1, \tau_1^1, \ldots, \tau_1^{n-3})$, $(\boldsymbol{y}_2, \tau_2, \tau_2^1, \ldots, \tau_2^{n-3})$ a curve in the plane starting from $\boldsymbol{y}_1$ and ending at $\boldsymbol{y}_2$ with orientation of the tangent vector and successive derivatives with respect to s and equal to $\tau_1, \tau_1^1, \ldots, \tau_1^{n-3}$ at the beginning and to $\tau_2, \tau_2^1, \ldots, \tau_2^{n-3}$ at the end. This exercise is relatively easy using polynomials and transforming the boundary conditions into linear equations over the coefficients of the polynomials. However, taking into account small-time controllability is a little bit more tricky. Reference [47.22] proposes a flatness-based steering method built on convex combinations of canonical curves.

47.4.2 Equivalence Between Chained-Formed and Feedback-Linearizable Systems

In the previous section, we proposed methods to steer feedback-linearizable control systems or systems that can be put into chained-form. We now give a necessary and sufficient condition for feedback-linearizability.

Feedback-Linearizability: A Necessary and Sufficient Condition

In [47.23], *Rouchon* gives conditions to check whether or not a system is feedback-linearizable. For two-input driftless systems a necessary and sufficient condition is the following: let us define as Δ^k, $k > 0$ the collection of distributions (i. e., set of vector fields) iteratively defined by: $\Delta_0 = \text{span}\{f_1, f_2\}$, $\Delta_1 = \text{span}\{f_1, f_2, [f_1, f_2]\}$ and $\Delta_{i+1} = \Delta_0 + [\Delta_i, \Delta_i]$ with $[\Delta_i, \Delta_i] = \text{span}\{[f, g], f \in \Delta_i, g \in \Delta_i\}$. A system with two-dimensional input is feedback-linearizable iff $\text{rank}(\Delta_i) = 2 + i$.

Example: The Chained-Form System

Let us consider the chained-form system defined by (47.3)–(47.6). The control vector fields of this system are

$$f_1 = (1, 0, z_2, \ldots, z_{n-1}) ,$$
$$f_2 = (0, 1, \ldots, 0) ,$$

$\text{rank}\Delta_0 = 2$. If we compute $f_3 = [f_1, f_2] = (0, 0, 1, 0, \ldots, 0)$, we note that $\text{rank}\Delta_1 = 3$. By computing $f_i = [f_1, f_{i-1}]$ for i up to n, we find a sequence $f_i = (0, \ldots, 0, 1, 0, \ldots, 0)$, where 1 is at position i. Therefore, $\text{rank}\Delta_i = 2 + i$ for i up to $n - 2$ and the chained-form system is feedback linearizable. This conclusion could have been drawn in a more straightforward way by noticing that the state can be reconstructed from (z_1, z_2) and its derivatives. (z_1, z_2) is thus the linearizing output of the chained-form system.

47.5 Robots and Trailers

Robotics systems studied in motion planning are mainly those composed of a mobile robot alone or towing one or several trailers. The input of these systems is two dimensional.

47.5.1 Differentially-Driven Mobile Robots

The simplest mobile robot, namely the differentially-driven mobile robot displayed in Fig. 47.1, is obviously feedback-linearizable. the trajectory of the center of the wheel axis (the linearizing output) of the robot gives the orientation of the robot. It can thus be steered using a flatness-based local steering method.

47.5.2 Differentially-Driven Mobile Robots Towing One Trailer

The differentially-driven mobile robot towing a trailer hitched on top of the wheel axis of the robot displayed in Fig. 47.3 is feedback linearizable and the linearizing output is the center of the wheel axis of the trailer. The differentially-driven mobile robot towing a trailer with hitched kingpin (Fig. 47.4) is also feedback lineariz-

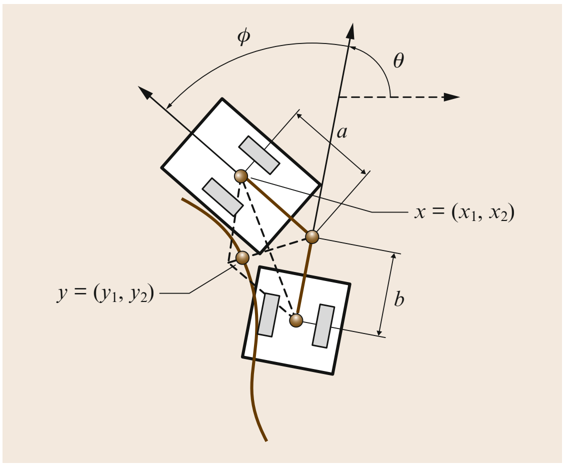

Fig. 47.4 A differentially-driven mobile robot towing a trailer with hitched kingpin is feedback linearizable

able [47.24] but the relation between linearizing output and configuration variables is more intricate

$$\begin{cases} y_1 = x_1 - b\cos\theta + L(\phi)\dfrac{b\sin\theta + a\sin(\theta+\phi)}{\sqrt{a^2+b^2+2ab\cos\phi}} , \\ y_2 = x_2 - b\sin\theta - L(\phi)\dfrac{a\cos(\theta+\phi) + b\cos\theta}{\sqrt{a^2+b^2+2ab\cos\phi}} , \end{cases}$$

where L is defined by the following elliptic integral

$$L(\phi) = ab\int_0^{\phi} \frac{\cos\sigma}{\sqrt{a^2+b^2+2ab\cos(\sigma)}}\mathrm{d}\sigma ,$$

which is the linearizing output of the system. These relations together with the following ones

$$\tan\tau = \frac{a\sin(\theta+\phi) + b\sin\theta}{b\cos\theta + a\cos(\theta+\phi)} ,$$

$$\kappa = \frac{\sin(\phi)}{\cos\phi\sqrt{a^2+b^2+2ab\cos\phi} + L(\phi)\sin(\phi)} ,$$

make it possible to reconstruct the configuration variables from the linearizing output and its two first derivatives.

47.5.3 Car-Like Mobile Robots

A car-like mobile robot is constituted of a fixed rear wheel axis and of two steerable front wheels, the axes of which intersect at the center of curvature (Fig. 47.5). A car-like mobile robot is kinematically equivalent to a differentially-driven mobile robot towing a trailer hitched on top of the wheel axis of the robot (Fig. 47.3): the virtual front wheel corresponds to the differentially-driven robot, while the body of the car corresponds to the trailer.

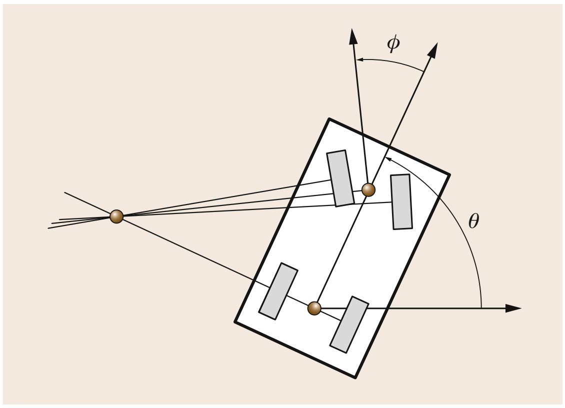

Fig. 47.5 Car-like mobile robot. The axes of the front wheels intersect at the center of curvature. The steering angle ϕ is the angle between the longitudinal axis of the car and a virtual front wheel in the middle of the two front wheels

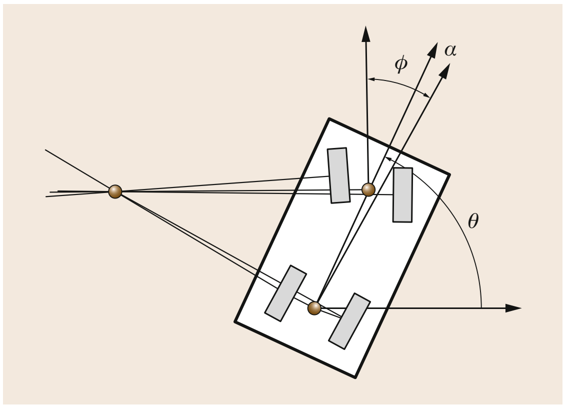

Fig. 47.6 Bi-steerable robot. The front and rear wheels are steerable and there is a relation between the rear steering angle and the front steering angle: $\alpha = f(\phi)$

47.5.4 Bi-Steerable Mobile Robots

The bi-steerable mobile robot (Fig. 47.6) is a car with front and rear steerable wheels and with a relation between the front and rear steering angles. This system has been proved to be feedback linearizable in [47.25]. As for the mobile robot towing a trailer with hitched kingpin, the linearizing output is a moving point in the robot reference frame.

47.5.5 Differentially-Driven Mobile Robots Towing Trailers

Let us consider the differentially-driven mobile robot towing a trailer connected on top of the wheel axis of the robot, and let us add an arbitrary number of trailers,

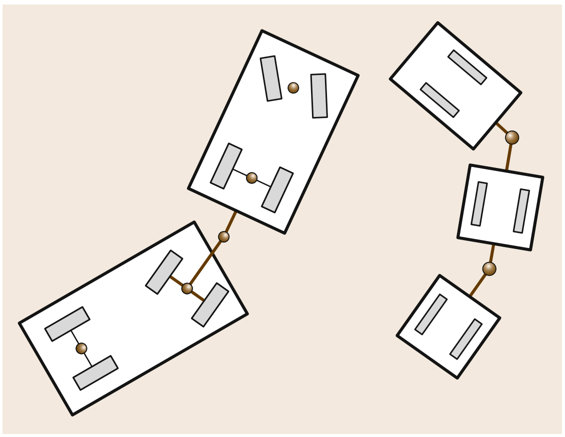
Fig. 47.7 A truck towing a towbar trailer with hitched kingpin and a differentially-driven mobile robot towing two trailers with hitched kingpin: two open problems for exact path planning for nonholonomic systems

each one connected on top of the wheel axis of the previous one. By differentiating once the curve followed by the center of the last trailer, we get the orientation of the last trailer (this orientation coincides with the orientation of the tangent to the curve). If we know the orientation and the position of the center of the last trailer along the path, we can reconstruct the curve followed by the center of the previous trailer. Repeating this reasoning, we can reconstruct the trajectory of the whole system by differentiating a sufficient number of times. The system is therefore feedback linearizable and the linearizing output is the center of the last trailer.

Combining the above reasoning with the systems reviewed in this section, we can build hybrid feedback-linearizable trailer systems. For instance, a differentially-driven mobile robot towing an arbitrary number n of trailers each one connected on top of the wheel axis of the previous trailer, except the last trailer with hitched kingpin is feedback linearizable. Simply consider the two last trailers as a feedback-linearizable system composed of a mobile robot with one trailer as in Fig. 47.4. The linearizing output of this system enables us to reconstruct the trajectory of the two last trailers. The center of trailer $n-1$ is a linearizing output for the mobile robot towing the $n-1$ last trailers.

47.5.6 Open Problems

Finally, all the systems for which we are able to plan exact motions between arbitrary pairs of configurations are included in the large class of feedback-linearizable systems. We have indeed seen that chained-form systems are also part of this class. For other systems, no exact solution has been proposed up to now. For instance, neither of the systems displayed in Fig. 47.7 is feedback linearizable. They do not satisfy the necessary condition of Sect. 47.4.2.

47.6 Approximate Methods

To deal with nonholonomic systems not belonging to any class of systems for which exact solutions exist, numerical approximate solutions have been developed. We review some of these methods in this section.

47.6.1 Forward Dynamic Programming

In [47.26] *Barraquand* and *Latombe* propose a dynamic programming approach to nonholonomic path planning. Admissible paths are generated by a sequence of constant input values, applied over a fixed interval of time δt. Starting from the initial configuration the search generates a tree: the children of a given configuration $\boldsymbol{q}$ are obtained by setting the input to a constant value and integrating the differential system over δt. The configuration space is discretized into an array of cells of equal size (i. e., hyper-parallelepipeds). A child $\boldsymbol{q}'$ of a configuration $\boldsymbol{q}$ is inserted in the search tree if and only if the computed path from $\boldsymbol{q}$ to $\boldsymbol{q}'$ is collision free and $\boldsymbol{q}'$ 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 has been proved to be asymptotically complete with respect 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 taking into account optimality criteria such as the path length or the number of reversals. Asymptotical optimality to generate the minimum of reversals is proved for the car-like robot alone.

47.6.2 Discretization of the Input Space

In [47.27] *Divelbiss* and *Wen* propose a method to produce an admissible collision-free path for a nonholonomic mobile robot in the presence of obstacles. They restrict the set of input functions over the subspace spanned by Fourier basis over interval $[0, 1]$. An input function is thus represented by a finite-dimensional vector λ. Reaching a goal configuration thus becomes a nonlinear system of equations, the unknowns of which are the coordinates λ_i's of λ. The authors use the Newton–Raphson method to find a solution. Obstacles are defined by inequality constraints over the configuration space. The path is discretized into N samples. The noncollision constraint expressed at these sample points yields the inequality constraint over vector λ. These inequality constraints are turned into equality constraints through function g defined as

$$g(c) = \begin{cases} (1 - \mathrm{e}^c)^2 & \text{if } c > 0\,, \\ 0 & \text{if } c \le 0\,. \end{cases}$$

Reaching a goal configuration while avoiding obstacles thus becomes a nonlinear system of equations over vector λ again solved using the Newton–Raphson method. The method is rather efficient for short maneuvers. The main difficulty is to tune the order of the Fourier expansion. Long motions in cluttered environments require a higher order, while motion in empty space can be solved with a low order. The authors do not mention the problem of numerical instability in the integration of the dynamic system.

47.6.3 Input-Based Rapidly Exploring Random Trees

The RRT algorithms described in Chap. 7 can be used without a local steering method to plan paths for nonholonomic systems. New nodes can be generated from existing nodes by applying random input functions over an interval of time [47.28]. The main difficulty consists in finding a distance function that really accounts for the distance the system needs to travel to go from one configuration to another. Moreover, the goal is never exactly reached. This latter drawback can be overcome by post-processing the path returned by RRT using a path deformation method, as described in [47.29].

47.7 From Motion Planning to Obstacle Avoidance

Up to now we have described motion planning techniques. Their objective is to compute a collision-free trajectory to the target configuration that complies with the vehicle constraints. They assume a perfect model of the robot and scenario. The advantage of these techniques is that they provide complete and global solutions of the problem. Nevertheless, when the surroundings are unknown and unpredictable, these techniques fail.

A complementary way to face the motion problem is obstacle avoidance. The objective is to move a vehicle towards a target location free of collisions with the obstacles collected by the sensors during motion execution. The advantage of reactive obstacle avoidance is to compute motion by introducing the sensor information within the control loop, used to adapt the motion to any contingency incompatible with initial plans.

The main cost of considering the reality of the world during execution is locality. In this instance, if global reasoning is required, a trap situation could occur. Despite this limitation, obstacle avoidance techniques are mandatory to deal with mobility problems in unknown and evolving surroundings.

Notice that methods have been developed to combine both the global point of view of motion planning and the local point of view of obstacle avoidance. How can we consider robot perception at the planning level? This is so-called sensor-based motion planning. Several variants exists, such as the Bug algorithms initially introduced in [47.30]. However, non of them consider the practical context of nonholonomic mobile robots.

47.8 Definition of Obstacle Avoidance

Let $\mathcal{A}$ be the robot (a rigid object) moving in the workspace $\mathcal{W}$, whose configuration space is $\mathcal{CS}$. Let $\boldsymbol{q}$ be a configuration, $\boldsymbol{q}_t$ this configuration in time t, $\mathcal{A}(\boldsymbol{q}_t) \in \mathcal{W}$ the space occupied by the robot in this configuration. In the vehicle there is a sensor, which in $\boldsymbol{q}_t$ measures a portion of the space $\mathcal{S}(\boldsymbol{q}_t) \subset \mathcal{W}$ identifying a set of obstacles $\mathcal{O}(\boldsymbol{q}_t) \subset \mathcal{W}$.

Let $\boldsymbol{u}$ be a constant control vector and $\boldsymbol{u}(\boldsymbol{q}_t)$ this control vector applied in $\boldsymbol{q}_t$ during time δt. Given $\boldsymbol{u}(\boldsymbol{q}_t)$, the vehicle describes a trajectory $\boldsymbol{q}_{t+\delta t} = f(\boldsymbol{u}, \boldsymbol{q}_t, \delta t)$, with $\delta t \ge 0$. Let $\mathcal{Q}_{t,T}$ be the set of configurations of the trajectory followed from $\boldsymbol{q}_t$ with $\delta t \in [0, T]$, a given time interval. $T > 0$ is called the *sampling period*. Let $F : \mathcal{CS} \times \mathcal{CS} \to \mathbb{R}^+$ be a function that evaluates the progress of one configuration to another.

47.8.1 Obstacle Avoidance Problem

Let $\boldsymbol{q}_{\text{target}}$ be a target configuration. Then, in time t_i the robot $\mathcal{A}$ is in $\boldsymbol{q}_{t_i}$, where a sensor measurement

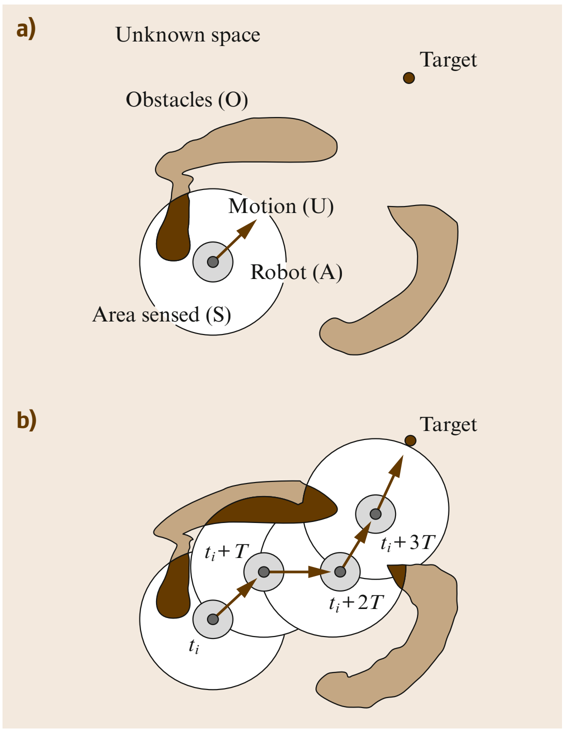

Fig. 47.8 **(a)** Obstacle avoidance problem consists of computing motion control that avoids collisions with the obstacles gathered by the sensors whilst driving the robot towards the target location. **(b)** The result of applying this technique at each time is a sequence of motions that drive the vehicle free of collisions to the target

is obtained $S(\boldsymbol{q}_{t_i})$, and thus an obstacle description $\mathcal{O}(\boldsymbol{q}_{t_i})$.

The objective is to compute a motion control $\boldsymbol{u}_i$, such that (i) the trajectory generated is free of collisions with the obstacles $\mathcal{A}(\mathcal{Q}_{t_i,T}) \bigcap \mathcal{O}(\boldsymbol{q}_{t_i}) = \emptyset$; and (ii) it makes the vehicle progress to the target location $F(\boldsymbol{q}_{t_i}, \boldsymbol{q}_{\text{target}}) < F(\boldsymbol{q}_{t_i+T}, \boldsymbol{q}_{\text{target}})$.

The result of solving this problem at each sample time (Fig. 47.8a) is a sequence of motion controls $\{\boldsymbol{u}_1 \ldots \boldsymbol{u}_n\}$ computed in execution time that avoids the obstacles gathered by the sensors, while making the vehicle progress towards the target location in each configuration $\{\boldsymbol{q}_{t_1} \ldots \boldsymbol{q}_{\text{target}}\}$ (Fig. 47.8b). Notice that the motion problem is a global problem. The obstacle avoidance methods are local and iterative techniques to address this problem. The disadvantage of locality (trap situations) is counteracted with the advantage of introducing the sensory information within the control cycle (to take into account the reality of the world in execution).

There are at least three aspects that affect the development of an obstacle avoidance method: the avoidance technique, the type of robot-sensor and the type of scenario. These subjects correspond to the next three sections. First, we describe the obstacle avoidance techniques (Sect. 47.9). Second, we discuss the techniques to adapt a given obstacle avoidance method to work on a vehicle taking into account the shape, kinematics, and dynamics (Sect. 47.10). The sensory processing is detailed in Chaps. 5 and 25 of this book. Finally, the usage of an obstacle avoidance technique on a vehicle in a given scenario is highly dependent on the scenario nature (static or dynamic, unknown or known, structured or not, or its size, for example). Usually, this problem is associated with the integration of planning and obstacle avoidance (Sect. 47.11).

47.9 Obstacle Avoidance Techniques

We describe here a taxonomy of obstacle avoidance techniques and some representative methods. First there are two groups, methods that compute the motion in one step and methods that do it in more than one. Methods of one step directly reduce the sensor information to motion control. There are two types:

- The *heuristic* methods were the first techniques used to generate motion based on sensors. The majority of this work was derived in classic planning methods and will not be described here. See [47.1, 30–33] for some representative work.
- The methods of *physical analogies* assimilate the obstacle avoidance to a known physical problem. We discuss here the potential field methods [47.34, 35]. Other works are variants adapted to uncertain models [47.36] or that use other analogies [47.37–39].

Methods with more than one step compute some intermediate information, which is processed next to obtain the motion:

- The methods of *subset of controls* compute an intermediate set of motion controls, and next they choose one of them as a solution. There are two types: (i) methods that compute a subset of motion directions. We describe here the vector field histogram (VFH) [47.40] and the obstacle restriction method [47.41] (see [47.42] for three-dimensional (3-D) workspaces). Another method is the steer angle field approach [47.43]. (ii) Methods that compute a subset of velocity controls. We describe here the dynamic window approach (DWA) [47.44] and velocity obstacles [47.45] (see [47.46] for moving obstacles). Another method based on similar principles but developed independently is the curvature velocity method [47.47].

- Finally, there are methods that compute some *high-level information* such as intermediate information, which is translated next in motion. We describe the basic versions of the nearness diagram navigation [47.48, 49]. The reader is directed to [47.50–52] for further improvements.

All the methods outlined here have advantages and disadvantages, depending on the navigation context, like uncertain worlds, motion at high speeds, motion in confined or troublesome spaces, etc. Unfortunately there is no metric available to quantitatively measure the performance of the methods. However, for an experimental comparison in terms of their intrinsic problems see [47.48].

47.9.1 Potential Field Methods

The potential field method (PFM) uses an analogy in which the robot is a particle that moves in the configuration space under the influence of a force field. While the target location exerts a force that attracts the particle $\boldsymbol{F}_{\text{att}}$, the obstacles exert repulsive forces $\boldsymbol{F}_{\text{rep}}$. At each time t_i, the motion is computed to follow the direction of the artificial force induced by the sum of both potentials $\boldsymbol{F}_{\text{tot}}(\boldsymbol{q}_{t_i}) = \boldsymbol{F}_{\text{att}}(\boldsymbol{q}_{t_i}) + \boldsymbol{F}_{\text{rep}}(\boldsymbol{q}_{t_i})$ (the most promising motion direction), Fig. 47.9.

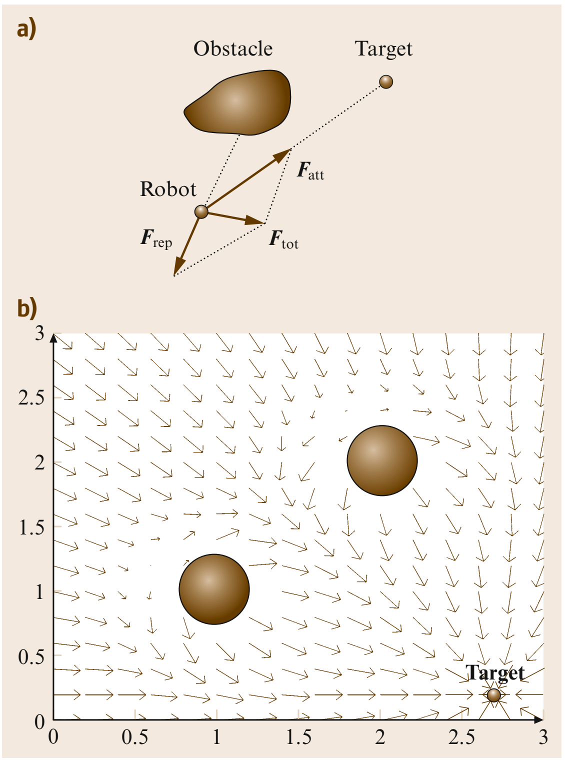
a)
Obstacle
Target
Robot
$\boldsymbol{F}_{\text{att}}$
$\boldsymbol{F}_{\text{rep}}$
$\boldsymbol{F}_{\text{tot}}$
b)
Target
0
0.5
1
1.5
2
2.5
3

Fig. 47.9 **(a)** Computation of the motion direction with a potential field method. The target attracts the particle $\boldsymbol{F}_{\text{att}}$ while the obstacle exerts a repulsive force $\boldsymbol{F}_{\text{rep}}$. The resulting force $\boldsymbol{F}_{\text{tot}}$ is the most promising motion direction. **(b)** Motion directions computed in each point of the space with the classic method

Example

$$\boldsymbol{F}_{\text{att}}(\boldsymbol{q}_{t_i}) = K_{\text{att}} \boldsymbol{n}_{q_{\text{target}}} \tag{47.8}$$

$$\boldsymbol{F}_{\text{rep}}(\boldsymbol{q}_{t_i}) = \begin{cases} K_{\text{rep}} \sum_j \left(\dfrac{1}{d(\boldsymbol{q}_{t_i}, \boldsymbol{p}_j)} - \dfrac{1}{d_0} \right) \boldsymbol{n}_{p_j} & \text{if } d(\boldsymbol{q}_{t_i}, \boldsymbol{p}_j) < d_0\,, \\ 0 & \text{otherwise} \end{cases} \tag{47.9}$$

where K_{att} and K_{rep} are the constants of the forces, d_0 is the influence distance of the obstacles $\boldsymbol{p}_j$, $\boldsymbol{q}_{t_i}$ is the current vehicle configuration and $\boldsymbol{n}_{q_{\text{target}}}$ and $\boldsymbol{n}_{p_j}$ are the unitary vectors that point from $\boldsymbol{q}_{t_i}$ to the target and each obstacle $\boldsymbol{p}_j$. From $\boldsymbol{F}_{\text{tot}}(\boldsymbol{q}_{t_i})$, the control $\boldsymbol{u}_i$ can be obtained with a position or force control [47.53].

This is the classic version, where the potentials only depend on the current vehicle configuration. Complementarily the generalized potentials depend also on the instantaneous robot velocity and vehicle accelerations.

Example

$$\boldsymbol{F}_{\text{rep}}(\boldsymbol{q}_{t_i}) = \begin{cases} K_{\text{rep}} \sum_j \left(\dfrac{a \dot{\boldsymbol{q}}_{t_i}}{[2ad(\boldsymbol{q}_{t_i}, \boldsymbol{p}_j) - \dot{\boldsymbol{q}}_{t_i}^2]} \right) \boldsymbol{n}_{p_j} \cdot \boldsymbol{n}_{\dot{q}_{t_i}} & \text{if } \dot{\boldsymbol{q}}_{t_i} > 0\,, \\ 0 \quad \text{otherwise} \end{cases} \tag{47.10}$$

where $\dot{\boldsymbol{q}}_{t_i}$ is the current robot velocity, $\boldsymbol{n}_{\dot{q}_{t_i}}$ the unitary vector pointing in the direction of the robot velocity, and a is the maximum vehicle acceleration. This expression arises when the repulsive potential is defined as the inverse of the difference between the estimated time until a collision takes place and the time needed to stop the robot until full backward acceleration. Notice how the repulsive only affect in the direction of motion of the vehicle, as opposed to the classic potential. For a comparison of the classic and generalized versions, and the relation with the way to compute the motion controls next, see [47.53]. This method is widely used because it is easy to understand and due to its clear mathematic formalism.

47.9.2 Vector Field Histogram

The vector field histogram (VFH) solves the problem in two steps by computing a set of candidate motion directions and then selecting one of them.

Candidate Set of Directions

Firstly the space is divided into sectors from the robot location. The method uses a polar histogram H constructed around the robot, where each component represents the obstacle polar density in the corresponding sector. The function that maps the obstacle distribution in sector k on the corresponding component of the histogram $h^k(\boldsymbol{q}_{t_i})$ is

$$h^k(\boldsymbol{q}_{t_i}) = \int_{\Omega_k} P(\boldsymbol{p})^n \cdot \left(1 - \frac{d(\boldsymbol{q}_{t_i}, \boldsymbol{p})}{d_{\max}}\right)^m \mathrm{d}\boldsymbol{p} \,. \qquad (47.11)$$

The dominion of integration is $\Omega_k = \{\boldsymbol{p} \in \mathcal{W} \backslash \boldsymbol{p} \in k \wedge d(\boldsymbol{q}_{t_i}, \boldsymbol{p}) < d_0\}$. The density $h^k(\boldsymbol{q}_{t_i})$ is proportional to the probability $P(\boldsymbol{r})$ that a point was occupied by an obstacle, and to a factor that increases when the distance to the point decreases.

Typically the resulting histogram has peaks (directions with high density of obstacles) and valleys (directions with low density). The set of candidate directions is the set of adjacent components with density lower than a given threshold, and closest to the component that contains the target direction. This set of components (sectors) is called the selected valley, and represents a set of candidate directions (Fig. 47.10).

Motion Computation

The objective of the next step is to choose a direction of this set. The strategy is to apply three heuristics that depend on the component that contains the target or on the size of the selected valley. The cases are checked in sequence:

1. Case 1: goal sector in the selected valley. Solution: $k_{\text{sol}} = k_{\text{target}}$, where k_{target} is the sector that contains the goal location.
2. Case 2: goal sector not in the selected valley and number of sectors of the valley greater than m. Solution: $k_{\text{sol}} = k_i \pm m/2$, where m is a fixed number of sectors and k_i the sector of the valley closer to the k_{target}.

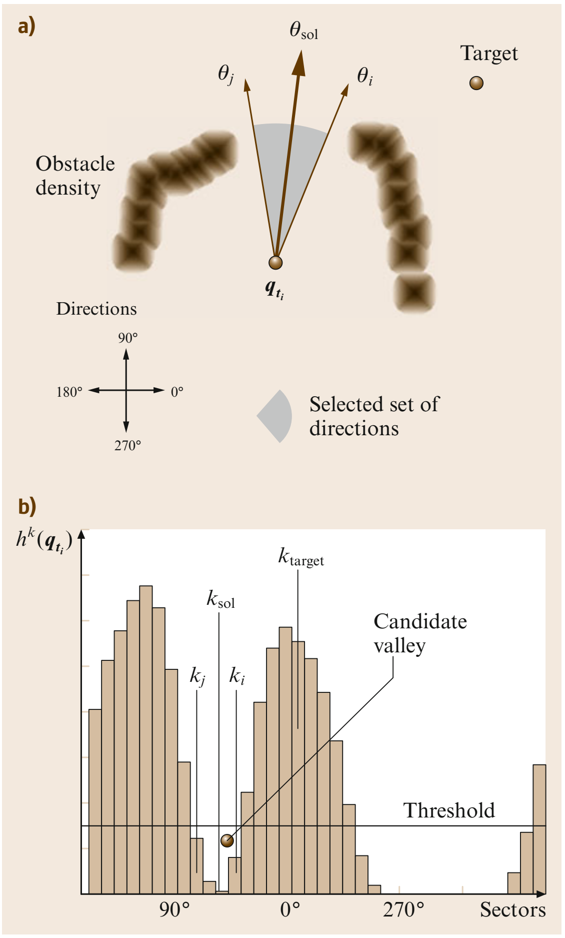

Fig. 47.10a,b Computation of the motion direction θ_{sol} with VFH. **(a)** Robot and obstacle occupancy distribution. **(b)** The candidate valley is the set of adjacent components with values lower than the threshold. The navigation case is Case 3, since the sector of the target k_{target} is not in the valley and the number of sectors is lower than a fixed quantity m ($m = 8$, i. e., 45°). Thus the solution is $k_{\text{sol}} = (k_i + k_j)/2$, whose bisector is θ_{sol} in **(a)**. The bisectors of k_i and k_j are θ_i and θ_j, respectively

3. Case 3: goal sector not in the selected valley and number of sectors of the valley lower or equal to m. Solution: $k_{\text{sol}} = (k_i + k_j)/2$, where k_i and k_j are the extreme sectors of the valley.

The result is a component or sector k_{sol}, whose bisector is the direction solution θ_{sol}. The velocity v_{sol} is inversely proportional to the distance to the closest obstacle. The control is $\boldsymbol{u}_i = (v_{\text{sol}}, \theta_{\text{sol}})$.

VFH is a method formulated to work with probability obstacle distributions, and thus, it is well adapted to work with uncertain sensors as ultrasonic sonars.

47.9.3 The Obstacle Restriction Method

The obstacle restriction method (ORM) solves the problem in three steps, where the result of the two first steps is a set of candidate motion directions. The first step is to compute an instantaneous subgoal if necessary. The second one associates a motion constraint to each obstacle and joins them next to compute the set of desirable directions. The last step is an strategy to compute the motion given this set.

Instantaneous Target Selection

This step computes a subgoal when is better to direct the motion towards a given zone of the space (which ameliorates the situation to reach the goal latter), rather than directly towards the goal itself. The subgoals are located in between obstacles or in the edge of an obstacle (Fig. 47.11a). Next, the process is to check with a local algorithm whether the goal can be reached from the robot location. If it is not, then the closest reachable subgoal to the goal is selected. To check if a point can be reached, there is a local algorithm that computes the existence of a local path that joins two locations [47.48].

Let $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ be two locations of the space, R the robot radius, and L a list of obstacle points, where $\boldsymbol{p}_i$ is an obstacle of the list. Let $\mathcal{A}$ and $\mathcal{B}$ be the two semiplanes divided by the line that joins $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$. Then, if for all the points of L, $d(\boldsymbol{p}_j, \boldsymbol{p}_k) > 2R$ (with $\boldsymbol{p}_j \in \mathcal{A}$ and $\boldsymbol{p}_k \in \mathcal{B}$), there is a collision free path that joins both locations. If this condition is not satisfied, then there is no local path (although a global one could exist). The interesting result is when the result is positive, since the guarantee that one point can be reached from the other exists.

The result of the process is the goal or an instantaneous subgoal (from now this location is called the target location). Note that this process is general and can be used as a preprocess step by other methods to instantaneously validate the goal location or to compute an instantaneous subgoal to drive the vehicle.

Candidate Set of Directions

For each obstacle i a set of not desirable directions for motion $S^i_{n\text{D}}$ is computed (motion constraint). This set is the union of two subsets S^i_1 and S^i_2. S^i_1 represents the side of the obstacle not suitable to do the avoidance and S^i_2 is an exclusion area around the obstacle (Fig. 47.11b). The motion constraint for the obstacle is the union of both sets $S^i_{n\text{D}} = S^i_1 \cup S^i_2$. The set of desired directions of motion is the complementary $S_\text{D} = \{[-\pi, \pi] \setminus S_{n\text{D}}\}$, where $S_{n\text{D}} = \cup_i S^i_{n\text{D}}$.

Motion Computation

The final step is to select a direction of motion. There are three cases that depend on the set of desirable directions S_D and on the target direction θ_target. The cases are checked in sequence:

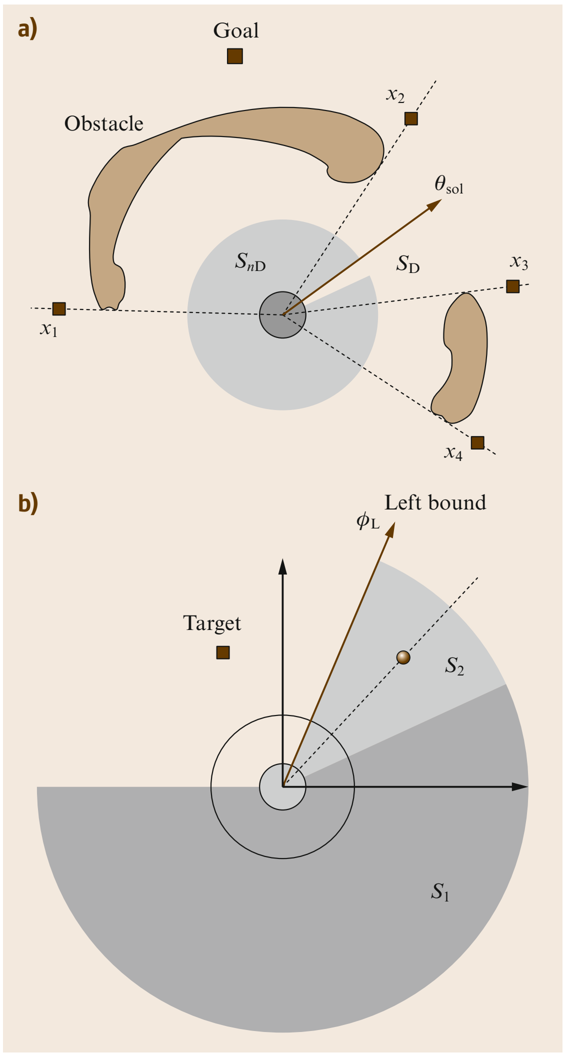

Fig. 47.11 (a) Distribution of the subgoals x_i. The selected instantaneous target location is x_2. The set of candidate directions is $S_{n\text{D}}$ and the solution is θ_sol (the second case). **(b)** The two sets of not desirable directions S_1 and S_2 for a given obstacle

1. Case 1: $S_\text{D} \neq \emptyset$ and $\theta_\text{target} \in S_\text{D}$. Solution: $\theta_\text{sol} = \theta_\text{target}$.
2. Case 2: $S_\text{D} \neq \emptyset$ and $\theta_\text{target} \notin S_\text{D}$. Solution: $\theta_\text{sol} = \theta_\text{lim}$, where θ_lim is the direction of S_D closest to θ_target.
3. Case 3: $S_\text{D} = \emptyset$. Solution: $\theta_\text{sol} = (\phi^l_\text{lim} + \phi^r_\text{lim})/2$, where ϕ^l_lim and ϕ^r_lim are the directions of S_{D_r} and S_{D_l} closer to θ_target (S_{D_r} and S_{D_l} are the set of desirable

directions of obstacles on the left and right-hand sides of the target, respectively).

The result is a direction of motion solution θ_{sol}. The velocity v_{sol} is inversely proportional to the distance to the closest obstacle. The control is $\boldsymbol{u}_i = (v_{\text{sol}}, \theta_{\text{sol}})$. This is a geometric-based method based on cases. The advantage is that it has been demonstrated to address effective motion in confined spaces.

47.9.4 Dynamic Window Approach

The dynamic window approach (DWA) (VIDEO 712) is a method that solves the problem in two steps, by computing as intermediate information a subset of the control space $\mathcal{U}$. For simplicity, we consider a motion control such as translational and rotational velocity (v, w); $\mathcal{U}$ is defined by

$$\begin{aligned}\mathcal{U} = \{\, &(v, w) \in \mathbb{R}^2 \setminus \\ &v \in [-v_{\max}, v_{\max}] \wedge \\ &w \in [-w_{\max}, w_{\max}]\,\} \,.\end{aligned} \tag{47.12}$$

Set of Candidate Controls

The candidate set of controls $\mathcal{U}_{\text{R}}$ contains the controls: (i) within the maximum velocities of the vehicle $\mathcal{U}$, (ii) that generate safe trajectories $\mathcal{U}_{\text{A}}$, and (iii) that can be reached within a short period of time given the vehicle accelerations $\mathcal{U}_{\text{D}}$. The set $\mathcal{U}_{\text{A}}$ contains the admissible controls. These controls can be canceled before collision by applying the maximum deceleration (a_v, a_w)

$$\mathcal{U}_{\text{A}} = \left\{(v, w) \in \mathcal{U} | v \leq \sqrt{2 d_{\text{obs}} a_v} \wedge w \leq \sqrt{2\theta_{\text{obs}} a_w}\right\} , \tag{47.13}$$

where d_{obst} and θ_{obst} are the distance to the obstacle and the orientation of the tangent to the trajectory in the obstacle. The set $\mathcal{U}_{\text{D}}$ contains the controls reachable in a short period

$$\begin{aligned}\mathcal{U}_{\text{D}} = \{\, &(v, w) \in \mathcal{U} \setminus \\ &v \in [v_o - a_v T, v_o + a_v T] \wedge \\ &w \in [w_o - a_w T, w_o + a_w T]\} \,,\end{aligned} \tag{47.14}$$

where $\dot{\boldsymbol{q}}_{t_i} = (v_o, w_o)$ is the current velocity.

The resulting subset of controls is (Fig. 47.12)

$$\mathcal{U}_{\text{R}} = \mathcal{U} \cap \mathcal{U}_{\text{A}} \cap \mathcal{U}_{\text{D}} \,. \tag{47.15}$$

Motion Computation

The next step is to select one control $\boldsymbol{u}_i \in \mathcal{U}_{\text{R}}$. The problem is set out as the maximization of an objective

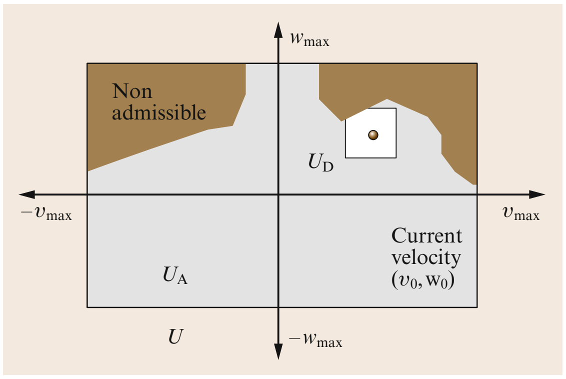

Fig. 47.12 Subset of controls $\mathcal{U}_{\text{R}} = \mathcal{U} \cap \mathcal{U}_{\text{A}} \cap \mathcal{U}_{\text{D}}$, where $\mathcal{U}$ contains the controls within the maximum velocities, $\mathcal{U}_{\text{A}}$ the admissible controls, and $\mathcal{U}_{\text{D}}$ the controls reachable in a short period of time

function

$$\begin{aligned}G(\boldsymbol{u}) = {}&\alpha_1 \cdot \text{Goal}(\boldsymbol{u}) + \alpha_2 \cdot \text{Clearance}(\boldsymbol{u}) \\ &+ \alpha_3 \cdot \text{Velocity}(\boldsymbol{u}) \,.\end{aligned} \tag{47.16}$$

This function is a compromise among Goal($\boldsymbol{u}$) that favors velocities that offer progress to the goal, Clerance($\boldsymbol{u}$) that favors velocities far from the obstacles, and Velocity($\boldsymbol{u}$) that favors high speeds. The solution is the control $\boldsymbol{u}_i$ that maximizes this function.

DWA solves the problem in the control space using information of the vehicle dynamics, thus the method is well adapted to work on vehicles with slow dynamic capabilities or that work at high speeds.

47.9.5 Velocity Obstacles

The method of velocity obstacles (VO) solves the problem in two steps, by computing as intermediate information a subset of the $\mathcal{U}$. The framework is equal to that of DWA. The difference is that the computation of the set of safe trajectories $\mathcal{U}_{\text{A}}$ takes into account the velocity of the obstacles, which is described next.

Let $\boldsymbol{v}_i$ be the velocity of obstacle i (which became enlarged with the vehicle radius occupying an area B_i) and $\boldsymbol{u}$ a given vehicle control. The set of colliding relative velocities is called *collision cone*

$$CC_i = \left\{\boldsymbol{u}_i | \lambda_i \bigcap B_i \neq \emptyset\right\} , \tag{47.17}$$

where λ_i is the direction of the unitary vector $\boldsymbol{u}_i = \boldsymbol{u}_i - \boldsymbol{v}_i$. The *velocity obstacle* is this set in a common absolute system of reference

$$VO_i = CC_i \oplus \boldsymbol{v}_i \,, \tag{47.18}$$

where $\oplus$ is the Minkowski vector sum. The set of unsafe trajectories is the union of the velocity obstacles for

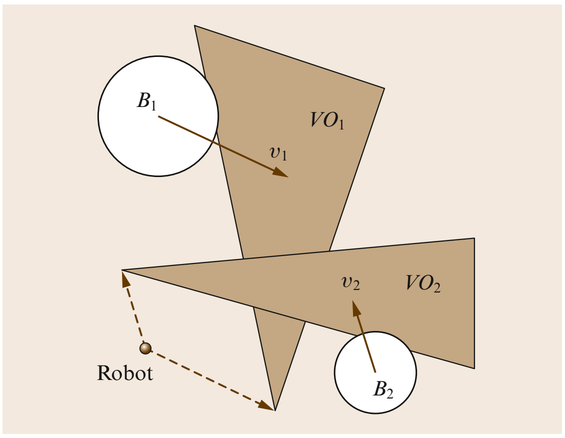

Fig. 47.13 Subset of not safe controls $\bar{\mathcal{U}}_A = VO_1 \cup VO_2$. A control vector out of this velocity generates a collision-free motion with the moving obstacles

each moving obstacle $\bar{\mathcal{U}}_A = \cup_i VO_i$ (Fig. 47.13). The advantage of this method is that takes into account the velocity of the obstacles; thus it is well suited in dynamic scenarios.

47.9.6 Nearness Diagram Navigation

This method is more a methodology to design obstacle avoidance methods rather than a method in itself. Nearness diagram navigation (ND) is an obstacle avoidance method obtained with a geometric implementation following this methodology. The idea behind is to employ a divide and conquer strategy based on situations to simplify the obstacle avoidance problem following the situated-activity paradigm [47.54]. First, there is a set of situations that represent all the cases among robot locations, obstacles, and target locations. Also, for each case there is a motion law associated with it. During the execution phase, at time t_i one situation is identified and the corresponding law is used to compute the motion.

Situations

The situations are represented in a binary decision tree. The selection of a situation depends on the obstacles $\mathcal{O}(\boldsymbol{q}_{t_i})$, on the robot location $\boldsymbol{q}_{t_i}$ and target $\boldsymbol{q}_{\text{target}}$. The criteria are based on high-level entities, like a security distance around the robot bounds and a motion area (that identifies suitable areas of motion). For example, one criterion is if there are obstacles within the security zone. Another is if the motion area is large or narrow. The result is only one situation, since by definition and representation (binary decision tree) the set of situations is complete and exclusive (Fig. 47.14).

Actions

Associated to any situation there is an action that computes the motion to adapt the behavior to the case

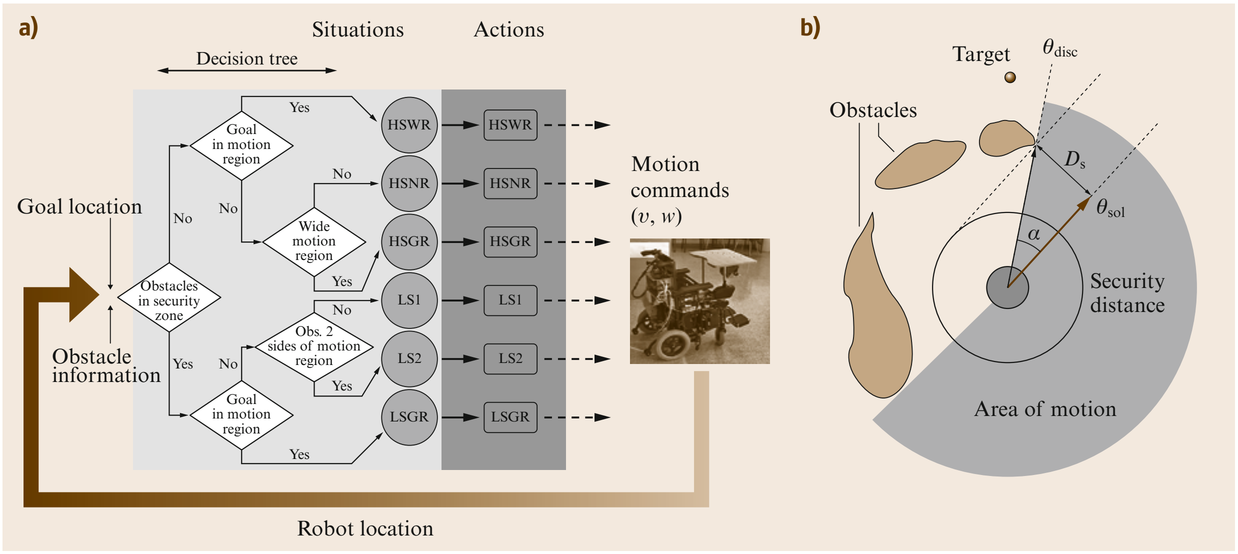

Fig. 47.14 **(a)** Design diagram of the method. Given the obstacle information and the target location, one situation is identified given some criteria. Next the associated action is executed computing the motion. **(b)** Example of computation of the solution of the ND (geometric implementation). The first step is to identify the situation. There are no obstacles closer than the security distance D_s. Next $\boldsymbol{q}_{\text{target}}$ is not within the motion area. Third, the motion area is wide. With these three criteria we identify the current situation, HSWR. In this situation, the associated action computes the control as $\boldsymbol{u}_i = (v_{\text{sol}}, \theta_{\text{sol}})$, where v_{sol} is the maximum velocity and θ_{sol} is computed as a deviation α from the limit direction θ_{disc} of the motion area closer to the target direction

represented by each situation. At a high level the actions describe the behavior desired in each situation. For example, one situation is when there are no obstacles within a security area and the goal is within the motion area (HSGR). The solution is to *move towards the goal*. Another situation is when there are no obstacles within a security area, and the goal is not within the motion area but is wide (HSWR). The solution is to *move towards the limit of the motion area but clearing the security area of obstacles*.

The interesting aspect of this method is that it employs a *divide and conquer* strategy to solve the navigation. As a consequence, it simplifies the difficulty of the problem. The first advantage is that the methodology is described at a symbolic level and, thus, there are many ways to implement the method. Second, a geometric implementation of this methodology (ND method) has been demonstrated to address difficult navigation cases, which translate in to achieve safe navigation in dense, complex, and difficult scenarios.

47.10 Robot Shape, Kinematics, and Dynamics in Obstacle Avoidance

There are three aspects of the vehicle that have to be taken into account during the obstacle avoidance process: shape, kinematics, and dynamics. The shape with the kinematics is a geometric problem that involves the representation of the vehicle configurations in collision, given the admissible trajectories $\mathcal{Q}_{t,\infty}$. The dynamics involves the accelerations and temporal considerations, and is derived in two aspects: (i) choosing a control reachable in a short period of time T given the current velocity $\dot{\boldsymbol{q}}_{t_i}$ and the maximum accelerations. (ii) Taking the braking distance into account, so that after a control execution, the vehicle can always stop before collision by applying the maximum deceleration (improving safety).

The problem of shape, kinematics, and dynamics in obstacle avoidance has been taken into account from three different points of view:

1. Designing a way to incorporate the constraints within the method (Sect. 47.9.4)
2. Developing techniques that abstract the vehicle aspects from the application of the method [47.55–58]
3. By techniques that break down the problem into subproblems and incorporate the aspects in sequence [47.59–61] after method usage.

47.10.1 Techniques that Abstract Vehicle Aspects

These techniques are based on constructing an abstraction layer between the aspects of the vehicle and the obstacle avoidance method, in such a way that when the method is applied its solutions already take into account these aspects [47.55–58]. We consider here vehicles whose elementary paths obtained under execution of constant controls can be approximated by circular arcs (for example, a differential-drive robot, syncro-drive, or a tri-cycle). To simplify, a control is translational and rotational velocity $\boldsymbol{u} = (v, w)$. The set of reachable controls $\mathcal{U}_{\mathrm{A}}$ given the current velocity $\dot{\boldsymbol{q}}_{t_i} = (v_0, w_0)$ and the maximum accelerations (a_v, a_w) is obtained by (47.14).

Abstraction Construction

For these vehicles, the configuration space CS is three-dimensional. The idea is to construct, centered in the robot at each time t_i, the manifold of the configuration space $ARM(\boldsymbol{q}_{t_i}) \equiv ARM$ defined by elementary circular paths. The function that defines the manifold is

$$\theta = f(x,y) = \begin{cases} \arctan 2\left(x, \dfrac{x^2-y^2}{2y}\right) & \text{if } y \geq 0\,, \\ -\arctan 2\left(x, -\dfrac{x^2-y^2}{2y}\right) & \text{otherwise}\,. \end{cases} \tag{47.19}$$

It is easy to see that function f is differentiable in $\mathbb{R}^2 \backslash (0,0)$. Thus $(x, y, f(x,y))$ defines a two-dimensional manifold in $\mathbb{R}^2 \times S^1$ when $(x,y) \in \mathbb{R}^2 \backslash (0,0)$. This manifold ARM contains all the configurations that can be reached at each step of the obstacle avoidance.

Next, in the ARM one computes the exact region of the configurations in collision CO_{ARM} given any shape of the robot (i. e., obstacle representation in the manifold). Given an obstacle point (x_p, y_p) and a point of the robot bounds (x_r, y_r), a point (x_s, y_s) of the CO_{ARM} boundary is obtained by

$$\begin{aligned} x_s &= (x_f + x_i) \cdot a\,, \\ y_s &= (y_f - y_i) \cdot a\,, \end{aligned} \tag{47.20}$$

with

$$\begin{aligned} a = {} & \left\{\left[\left(y_f^2 - y_i^2\right) + \left(x_f^2 - x_i^2\right)\right]\left[(y_f - y_i)^2 + (x_f - x_i)^2\right]\right\} \\ & \times \left[(y_f - y_i)^4 + 2\left(x_f^2 + x_i^2\right)(y_f - y_i)^2 \right. \\ & \left. + \left(x_f^2 - x_i^2\right)^2\right]^{-1}. \end{aligned}$$

This result is used to map the robot bounds for all obstacles in the manifold, computing the exact shape of the CO_{ARM}. Next, one compute the nonadmissible configurations CNA_{ARM} in the manifold ARM, which correspond to configurations that once reached at a given

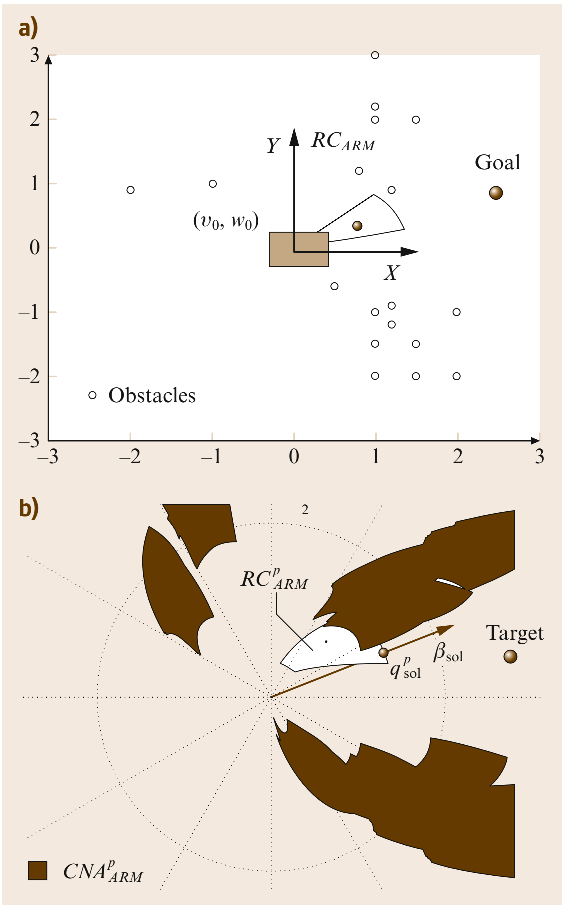

Fig. 47.15 (a) Robot location, obstacle information and goal location. **(b)** The abstraction layer in ARM^p. In this representation the obstacles are the nonadmissible region CNA^p_{ARM} in ARM and that the motion is omnidirectional (applicability conditions of many obstacle avoidance methods). The method is applied to obtain the most promising direction β_{sol}, used to obtain the configuration solution $\boldsymbol{q}^p_{sol}$ in the set of reachable configurations RC^p_{ARM}. Finally, the solution is the control $\boldsymbol{u}_{sol}$ that reaches this configuration at time T. This control complies with the kinematics and dynamics and takes into account the exact vehicle shape

velocity in time T, the vehicle cannot be stopped by applying the maximum deceleration without collision (i. e., there is not enough braking distance). These regions CNA_{ARM} are the CO_{ARM} *enlarged* by a magnitude that depends on the maximum vehicle accelerations. The set of configurations RC_{ARM} reachable by controls in a short period of time $\mathcal{U}_A$ is also computed in the ARM. Finally, a change of coordinates is applied to ARM leading to ARM^p. Its effect is that the elementary circular paths become straight segments in the manifold. As a consequence, the problem now is to move an omnidirectional point in a bidimensional space free of any constraint.

Method Application

The final step is to apply the avoidance method on the ARM^p to avoid the CNA^p_{ARM} regions. The method solution β_{sol} is the most promising direction in ARM^p. This direction is used to select and admissible location $\boldsymbol{q}^p_{sol} \notin CNA^p_{ARM}$ in the set of dynamic reachable configurations RC^p_{ARM}. Finally, the control $\boldsymbol{u}_{sol}$ that leads to this location at time T is selected. By construction, this control is kinematically and dynamically admissible, avoids collisions with the exact vehicle shape, and takes into account the braking distance (Fig. 47.15).

Note how with these techniques, the three-tridimensional obstacle avoidance problem is converted into the simple problem of moving a point in a two-dimensional space without kinematic and dynamic restrictions (applicability conditions of many existing methods). Thus, many existing methods can be applied in this abstraction, and as a consequence, the solutions take the vehicle constraints into account.

47.10.2 Techniques of Decomposition in Subproblems

These techniques address the obstacle avoidance problem taking into account the vehicle constraints by breaking down this problem into subproblems:

1. Obstacle avoidance
2. Kinematics and dynamics
3. The shape.

Each subproblem is dealt with in sequence (Fig. 47.16).

Obstacle Avoidance

First the obstacle avoidance method is used by assuming a circular and omnidirectional vehicle. The solution is the most promising motion direction and the velocity $\boldsymbol{u}_1 = (v_1, \theta_1)$ to direct the vehicle towards the target.

Kinematics and Dynamics

Second, this control is converted into a control that complies with the kinematics and dynamics, which tends to align the vehicle with the instantaneous motion direction θ_1 of $\boldsymbol{u}_1$. For example, [47.60, 61] modify the output of the obstacle avoidance method by a feedback action that aligns the vehicle with the direction solution in a least squares fashion. Moreover, [47.59] use a dynamic controller of the vehicle. This controller models the behavior of the vehicle as it would be pulled by a virtual force, and computes the mo-

Steps	Step 1	Step 2	Step 3
Robot aspects	Circular omnidirectional	Circular with kinematics and dynamics	Rectangular with kinematics and dynamics
Subproblem (solution)	Obstacle avoidance →	Motion controller →	Shape corrector

Fig. 47.16 Decomposition of the obstacle avoidance problem into subproblems that are addressed in steps, successively incorporating the vehicle aspects

tion that would result after applying this force during a short period of time. The motion generated is the new control. In order to use the controller, the previous control $\boldsymbol{u}_1$ is converted into an instantaneous force $\boldsymbol{F} = (\theta_1, F_{\max}(v_1/v_{\max}))$ input of the controller, which computes a control $\boldsymbol{u}_2$ that complies with the kinematic and dynamics.

Shape

The final step is to assure that the control $\boldsymbol{u}_2$ obtained in the previous step avoids collisions with the exact shape of the vehicle. To do this, a shape corrector is used to check collisions by dynamic simulation of the control. If there is collision, there is a search in the set of reachable controls (47.14) until one of them is collision free. The result of this process is a motion control $\boldsymbol{u}_i$ that guarantees obstacle avoidance and complies with the kinematics and dynamics of the vehicle.

47.11 Integration Planning – Reaction

In this section we show how the obstacle avoidance methods are integrated in real systems. On the one hand, the obstacle avoidance methods are local techniques to address the motion problem. Thus, they are doomed to fall in local minima that translate in trap situations or cyclic motions. This reveals the necessity of a more global reasoning. On the other hand, motion planning techniques compute a geometric path free of collisions, which guarantees global convergence. However, when the scenarios are unknown and evolve, these techniques fail, since the precomputed paths will almost surely collide with obstacles. It seems clear that one key aspect to build a motion system is to combine the best of both worlds: the global knowledge given by motion planning and the reactivity of the obstacle avoidance methods.

The most extended ways to specify the interaction between the deliberation and reaction are: (i) to precompute a path to the target deformed in execution as a function of the changes in the scenario obtained from the sensor information (systems of path deformation), for example, [47.62–67]. (ii) To use a planner at a high frequency with a tactical role, leaving the degree of execution to the reactor [47.68–73].

47.11.1 Systems of Path Deformation

Elastic bands is a method that initially assumes the existence of a geometric path to the target location (computed by a planner). The path is assimilated with a band, subjected to two types of forces: an inner contraction force and an external force. The inner force simulates the tension of a strip and maintains the stress. The external force is exerted by the obstacles and moves the band far from them. During the execution, the new obstacles produce forces that remove the band far from them, guaranteeing their avoidance. These methods are described in Chap. 37.

An extension of the method of path deformation was proposed in [47.64] for nonholonomic systems (VIDEO 80). Even though the objective is the same as for mobile robots without kinematic constraints, avoiding obstacle while following a trajectory, the concepts are completely different. The trajectory Γ of a nonholonomic system is completely defined by an initial configuration $\Gamma(0)$ and the value of the input function $\boldsymbol{u} \in C^1(I, \mathbb{R}^m)$, where I is an interval. The trajectory deformation method for nonholonomic systems is thus based on the perturbation of the input function of the current trajectory in order to achieve three objectives:

1. Keeping the nonholonomic constraints satisfied
2. Getting away from obstacles detected online by onboard sensors
3. Keeping unchanged the initial and last configurations of the trajectory after deformation.

Perturbing the input function of Γ, by a vector-valued input perturbation $\boldsymbol{v} \in C^1(I, \mathbb{R}^m)$, yields a trajectory deformation $\eta \in C^1(I, \mathbb{R}^n)$

$$\boldsymbol{u} \leftarrow \boldsymbol{u} + \tau \boldsymbol{v} \Rightarrow \Gamma \leftarrow \Gamma + \tau \eta ,$$

where τ is a positive, asymptotically small real number. As an approximation of order 1, the relation between $\boldsymbol{u}$ and η is given by the linearized system, which we do not express here.

Potential Field of Obstacles

Obstacles are taken into account by defining a potential field over the configuration space, which increases when the robot gets closer to obstacles. This potential field is lifted into the space of trajectories by integration along the trajectory of the configuration potential value.

Discretization of the Input Space

The space $C^1(I,\mathbb{R}^m)$ of input perturbations is an infinite-dimensional vector space. The choice of an input perturbation is restricted to a finite-dimensional subspace spanned by p arbitrary test functions, $\boldsymbol{e}_1,\ldots,\boldsymbol{e}_p$ where p is a positive integer. An input perturbation: $\boldsymbol{u}=\sum_{i=1}^{p}\lambda_i\boldsymbol{e}_i$ is thus defined by a vector $\lambda\in\mathbb{R}^p$. The variation of the trajectory potential is linear with respect to λ.

Boundary Conditions

The boundary conditions consisting in applying a trajectory perturbation equal to zero at both ends of interval I is linear with respect to λ. It is, therefore, easy to find a vector λ making the potential decrease and satisfying the boundary conditions.

Nonholonomic Constraint Deviation

The approximation of order 1 in the relation between the input perturbation and trajectory deformation induces a side effect: after a few iterations, the nonholonomic constraints are no longer satisfied. This side effect is corrected by considering an augmented system with n control vector fields $f_1,\ldots,f_n$ spanning $\mathbb{R}^n$ for each configuration and by keeping the input components u_{m+1} to u_n along the additional vector fields as close as possible to 0. Figure 47.17 shows an example of the trajectory deformation algorithm applied to a nonholonomic system.

47.11.2 Systems of Tactical Planning

The systems of tactical planning recompute at a high frequency a path to the target location, and use the main course to advise the obstacle avoidance module. The design of these motion systems involves at least the synthesis of three functionalities. The construction of a model, the deliberative planning, and the obstacle avoidance. The modeler constructs a representation base for deliberation and memory for the reactive behavior. The planner generates global plans used tactically to guide the obstacle avoidance module, which generates the local motion. Next we give a perspective of the three functionalities and three possible tools to implement them [47.73].

Model Builder Module

Construction of a model of the environment (to increase the spatial domain of the planning and used as local memory for obstacle avoidance). One possibility is to use a binary occupancy grid that is updated whenever a new sensory measurement is available, and to employ a scan matching technique [47.74, 75] to improve the vehicle odometry before integrating any new measure in the grid.

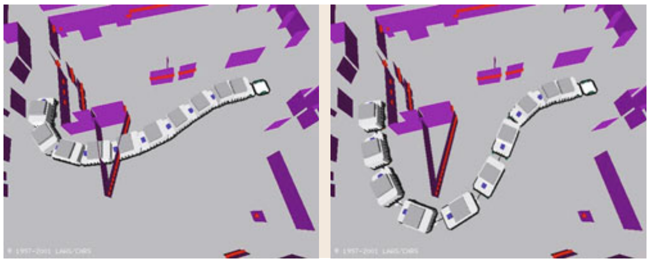
Fig. 47.17 A differentially-driven mobile robot towing a trailer applying the trajectory deformation algorithm to avoid obstacles detected online

Planner Module

Extraction of the connectivity of the free space (used to avoid the cyclical motions and trap situations). A good choice is a dynamic navigation function such as the D* [47.76, 77]. The idea behind the planner is to focus the search locally in the areas where the changes in the scenario structure have occurred and affect the computation of the path. The planner avoids the local minima and is computationally very efficient for real-time implementations.

Obstacle Avoidance Module

Computation of the collision-free motion. Any of the methods described in this chapter could be used. One possibility is the ND method (Sect. 47.9.6), since it has been demonstrated to be very effi-

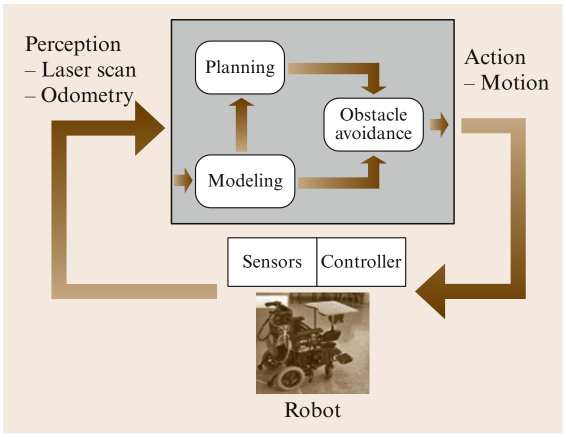

Fig. 47.18 Overview of the system of motion

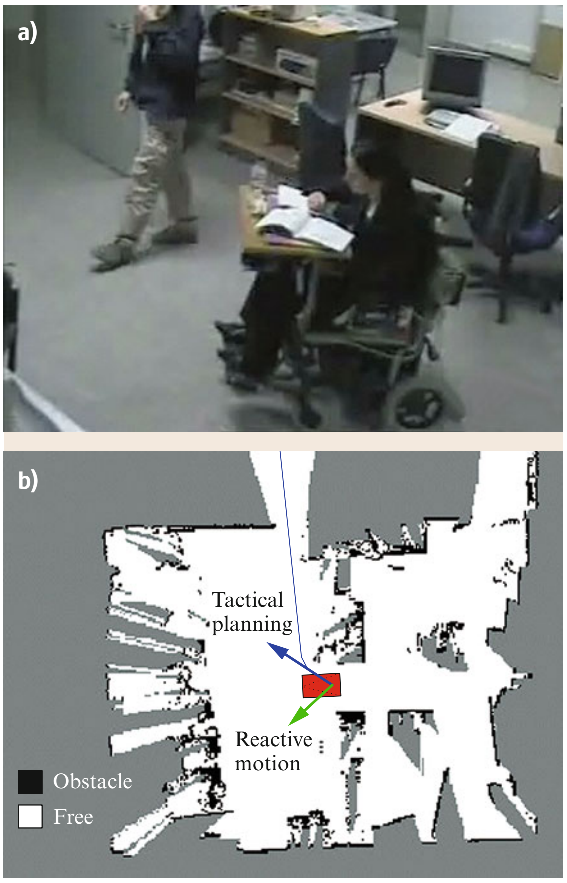

Fig. 47.19 **(a)** Snapshot of an experiment carried out with the motion system described working on a wheelchair robot equipped with a planar range laser sensor. **(b)** Information of the motion system at a given time: current accumulated map of the scenario, path computed with the planner and tactical direction of motion, and solution of the reactive obstacle avoidance method ◀

cient and robust in environments with little space to manoeuvre.

Globally the system works as follows (Fig. 47.18): given a laser scan and the odometry of the vehicle, the model builder incorporates this information into the existing model. Next, the information of the changes in obstacle and free space in the model is used by the planner module to compute the course to follow to reach the goal. Finally, the avoidance module uses the information of the obstacles contained in the grid and information of this tactical planner to generate the motion (to drive the vehicle free of collisions towards the goal). The motion is executed by the vehicle controller and the process restarts with a new sensorial measurement. It is important to stress that the three modules should work synchronously within the perception – action cycle for consistency reasons. The advantage of these systems is that the synergy of the modules allows to avoid trap situations and the cyclic motions (limitations associated to the local nature of the avoidance methods). Figure 47.19 shows a snapshot of an experiment carried out with this motion system.

47.12 Conclusions, Future Directions, and Further Reading

The algorithmic tools presented in this chapter show that motion planning and obstacle avoidance research techniques have reached a level of maturity that allow their transfer onto real platforms (VIDEO 710, VIDEO 714). Today, several indoor mobile robots use obstacle avoidance techniques on a daily basis to guide visitors in museums. Outdoor applications still require some developments in perception and modeling. For these applications, 3-D sensing capabilities are necessary to model the environment but also to detect obstacles. As an example, several car manufacturer are working on parallel parking assistance. The difficult point of this application is to build a model of a parking spot from 3-D data in a great variety of environments.

The main challenge of autonomous motion in mobile robots currently consists in integrating techniques from different robotics research domains in order to plan and execute motions for very complex systems, like humanoid robots, for instance. Integration in the sense that different software components need to work together on a machine but also in a scientific meaning: the classical motion planning formulation with configuration variables that locate the robot with respect to a global reference frame does not fit partially known environments with imprecise maps. Robotic tasks like motion need to be specified with respect to landmarks of the environment. For instance, grasping an object is by definition a motion specified with respect to the position of the object. Very little has been done to design a general framework in this direction.

For useful complementary reading on motion planning and obstacle avoidance for mobile robots see [47.5, 78–80].

Video-References

VIDEO 80 Sensor-based trajectory deformation and docking for nonholonomic mobile robots available from http://handbookofrobotics.org/view-chapter/47/videodetails/80

VIDEO 707 Autonomous robotic smart wheelchair navigation in an urban environment available from http://handbookofrobotics.org/view-chapter/47/videodetails/707

VIDEO 708 Sena wheelchair: Autonomous navigation at University of Malaga (2007) available from http://handbookofrobotics.org/view-chapter/47/videodetails/708

VIDEO 709 Robotic wheelchair: Autonomous navigation with Google Glass available from http://handbookofrobotics.org/view-chapter/47/videodetails/709

VIDEO 710 A ride in the google self driving car available from http://handbookofrobotics.org/view-chapter/47/videodetails/710

VIDEO 711 Mobile robot navigation system in outdoor pedestrian environment available from http://handbookofrobotics.org/view-chapter/47/videodetails/711

VIDEO 712 Mobile robot autonomous navigation in Gracia district, Barcelona available from http://handbookofrobotics.org/view-chapter/47/videodetails/712

VIDEO 713 Autonomous navigation of a mobile vehicle available from http://handbookofrobotics.org/view-chapter/47/videodetails/713

VIDEO 714 Autonomous robot cars drive DARPA Urban challenge available from http://handbookofrobotics.org/view-chapter/47/videodetails/714

References

47.1 N.J. Nilson: A mobile automaton: An application of artificial intelligence techniques, 1st Int. Jt. Conf. Artif. Intell. (1969) pp. 509–520

47.2 A. Thompson: The navigation system of the JPL robot, 5th Int. Jt. Conf. Artif. Intell., Cambridge (1977) pp. 749–757

47.3 G. Giralt, R. Sobek, R. Chatila: A multi-level planning and navigation system for a mobile robot: A 1st approach to Hilare, 6th Int. Jt. Conf. Artif. Intell. (1979) pp. 335–337

47.4 T. Lozano-Pérez: Spatial planning: A configuration space approach, IEEE Trans. Comput. **32**(2), 108–120 (1983)

47.5 J.C. Latombe: *Robot Motion Planning* (Kluwer, Dordrecht 1991)

47.6 J.-P. Laumond: Feasible trajectories for mobile robots with kinematic and environment constraints. In: *Intelligent Autonomous Systems*, ed. by F.C.A. Groen (Hertzberger, Amsterdam 1987) pp. 346–354

47.7 Z. Li, J.F. Canny: *Nonholonomic Motion Planning* (Kluwer, Dordrecht 1992)

47.8 M. Likhachev, D. Ferguson: Planning long dynamically-feasible maneuvers for autonomous vehicles, Int. J. Robotics Res. **28**(8), 933–945 (2009)

47.9 H. Sussmann: Lie brackets, real analyticity and geometric control. In: *Differential Geometric Control Theory*, Progress in Mathematics, Vol. 27, ed. by R. Brockett, R. Millman, H. Sussmann (Birkhauser, New York 1982) pp. 1–116

47.10 H.J. Sussmann, V. Jurdjevic: Controllability of nonlinear systems, J. Differ. Equ. **12**, 95–116 (1972)

47.11 J.-P. Laumond: Singularities and topological aspects in nonholonomic motion planning. In: *Nonholonomic motion Planning*, ed. by Z. Li, J.F. Canny (Kluwer, Boston 1992) pp. 149–199

47.12 J.-P. Laumond, J.J. Risler: Nonholonomic systems: Controllability and complexity, Theor. Comput. Sci. **157**, 101–114 (1996)

47.13 J.-P. Laumond, P. Jacobs, M. Taix, R. Murray: A motion planner for nonholonomic mobile robot, IEEE Trans. Robotics Autom. **10**(5), 577–593 (1994)

47.14 P. Cheng, S.M. LaValle: Reducing metric sensitivity in randomized trajectory design, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2001) pp. 43–48

47.15 J.A. Reeds, R.A. Shepp: Optimal paths for a car that goes both forward and backwards, Pac. J. Math. **145**(2), 367–393 (1990)

47.16 P. Souères, J.-P. Laumond: Shortest path synthesis for a car-like robot, IEEE Trans. Autom. Control **41**(5), 672–688 (1996)

47.17 D. Balkcom, M. Mason: Time optimal trajectories for bounded velocity differential drive vehicles, Int. J. Robotics Res. **21**(3), 199–218 (2002)

47.18 D. Tilbury, R. Murray, S. Sastry: Trajectory generation for the *n*-trailer problem using Goursat normal form, IEEE Trans. Autom. Control **40**(5), 802–819 (1995)

47.19 S. Sekhavat, J.-P. Laumond: Topological property for collision-free nonholonomic motion planning: The case of sinusoidal inputs for chained form systems, IEEE Trans. Robotics Autom. **14**(5), 671–680 (1998)

47.20 M. Fliess, J. Lévine, P. Martin, P. Rouchon: Flatness and defect of non-linear systems: Introductory theory and examples, Int. J. Control **61**(6), 1327–1361 (1995)

47.21 P. Rouchon, M. Fliess, J. Lévine, P. Martin: Flatness and motion planning: The car with *n* trailers, Eur. Control Conf. (1993) pp. 1518–1522

47.22 F. Lamiraux, J.-P. Laumond: Flatness and small-time controllability of multibody mobile robots:

Application to motion planning, IEEE Trans. Autom. Control **45**(10), 1878–1881 (2000)
47.23 P. Rouchon: Necessary condition and genericity of dynamic feedback linearization, J. Math. Syst. Estim. Control **4**(2), 1–14 (1994)
47.24 P. Rouchon, M. Fliess, J. Lévine, P. Martin: Flatness, motion planning and trailer systems, IEEE Int. Conf. Decis. Control (1993) pp. 2700–2705
47.25 S. Sekhavat, J. Hermosillo: Cycab bi-steerable cars: A new family of differentially flat systems, Adv. Robotics **16**(5), 445–462 (2002)
47.26 J. Barraquand, J.C. Latombe: Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles, Algorithmica **10**, 121–155 (1993)
47.27 A. Divelbiss, T. Wen: A path space approach to nonholonomic motion planning in the presence of obstacles, IEEE Trans. Robotics Autom. **13**(3), 443–451 (1997)
47.28 S. LaValle, J. Kuffner: Randomized kinodynamic planning, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1999) pp. 473–479
47.29 F. Lamiraux, E. Ferré, E. Vallée: Kinodynamic motion planning: Connecting exploration trees using trajectory optimization methods, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2004) pp. 3987–3992
47.30 V. Lumelsky, A. Stepanov: Path planning strategies for a point mobile automation moving admist unknown obstacles of arbitrary shape, Algorithmica **2**, 403–430 (1987)
47.31 R. Chatila: Path planning and environmental learning in a mobile robot system, Eur. Conf. Artif. Intell. (1982)
47.32 L. Gouzenes: Strategies for solving collision-free trajectories problems for mobile robots and manipulator robots, Int. J. Robotics Res. **3**(4), 51–65 (1984)
47.33 R. Chattergy: Some heuristics for the navigation of a robot, Int. J. Robotics Res. **4**(1), 59–66 (1985)
47.34 O. Khatib: Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robotics Res. **5**, 90–98 (1986)
47.35 B.H. Krogh, C.E. Thorpe: Integrated path planning and dynamic steering control for autonomous vehicles, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1986) pp. 1664–1669
47.36 J. Borenstein, Y. Koren: Real-time obstacle avoidance for fast mobile robots, IEEE Trans. Syst. Man Cybern. **19**(5), 1179–1187 (1989)
47.37 K. Azarm, G. Schmidt: Integrated mobile robot motion planning and execution in changing indoor environments, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1994) pp. 298–305
47.38 A. Masoud, S. Masoud, M. Bayoumi: Robot navigation using a pressure generated mechanical stress field, the biharmonical potential approach, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1994) pp. 124–129
47.39 L. Singh, H. Stephanou, J. Wen: Real-time robot motion control with circulatory fields, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1996) pp. 2737–2742
47.40 J. Borenstein, Y. Koren: The vector field histogram-fast obstacle avoidance for mbile robots, IEEE Trans. Robotics Autom. **7**, 278–288 (1991)
47.41 J. Minguez: The obstacle restriction method (ORM): Obstacle avoidance in difficult scenarios, IEEE/RSJ Int. Conf. Intell. Robot Syst. (IROS) (2005)
47.42 D. Vikerimark, J. Minguez: Reactive obstacle avoidance for mobile robots that operate in confined 3-D workspaces, IEEE Mediterr. Electrotech. Conf. (2006)
47.43 W. Feiten, R. Bauer, G. Lawitzky: Robust obstacle avoidance in unknown and cramped environments, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1994) pp. 2412–2417
47.44 D. Fox, W. Burgard, S. Thrun: The dynamic window approach to collision avoidance, IEEE Robotics Autom. Mag. **4**(1), 23–33 (1997)
47.45 P. Fiorini, Z. Shiller: Motion planning in dynamic environments using velocity obstacles, Int. J. Robotics Res. **17**(7), 760–772 (1998)
47.46 F. Large, C. Laugier, Z. Shiller: Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles, Auton. Robots **19**(2), 159–171 (2005)
47.47 R. Simmons: The curvature-velocity method for local obstacle avoidance, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1996) pp. 3375–3382
47.48 J. Minguez, L. Montano: Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios, IEEE Trans. Robotics Autom. **20**(1), 45–59 (2004)
47.49 J. Minguez, J. Osuna, L. Montano: A divide and conquer strategy to achieve reactive collision avoidance in troublesome scenarios, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2004)
47.50 J.W. Durham, F. Bullo: Smooth nearness-diagram navigation, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2008)
47.51 C.-C. Yu, W.-C. Chen, C.-C. Wang: Self-tuning nearness diagram navigation, Int. Conf. Serv. Interact. Robotics (SIRCon) (2009)
47.52 M. Mujahad, D. Fischer, B. Mertsching, H. Jaddu: Closest gap based (CG) reactive obstacle avoidance Navigation for highly cluttered environments, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2010) pp. 1805–1812
47.53 R.B. Tilove: Local obstacle avoidance for mobile robots based on the method of artificial potentials, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1990) pp. 566–571
47.54 R.C. Arkin: *Behavior-Based Robotics* (MIT Press, Cambridge 1999)
47.55 J. Minguez, L. Montano: Abstracting any vehicle shape and the kinematics and dynamic constraints from reactive collision avoidance methods, Eur. Conf. Mob. Robots (2007)
47.56 J. Minguez, L. Montano, J. Santos-Victor: Abstracting the vehicle shape and kinematic constraints from the obstacle avoidance methods, Auton. Robots **20**(1), 43–59 (2006)
47.57 L. Armesto, V. Girbés, M. Vincze, S. Olufs, P. Muñoz-Benavent: Mobile robot obstacle avoidance based

on quasi-holonomic smooth paths, Lect. Notes Comput. Sci. **7429**, 244–255 (2012)

47.58 J.L. Blanco, J. González, J.A. Fernández-Madrigal: Foundations of parameterized trajectories-based space transformations for obstacle avoidance. In: *Motion Planning*, ed. by X.-Y. Jing (InTech, Rijeka 2008)

47.59 J. Minguez, L. Montano: Robot navigation in very complex dense and cluttered indoor/outdoor environments, 15th IFAC World Congr. (2002)

47.60 A. De Luca, G. Oriolo: Local incremental planning for nonholonomic mobile robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1994) pp. 104–110

47.61 A. Bemporad, A. De Luca, G. Oriolo: Local incremental planning for car-like robot navigating among obstacles, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1996) pp. 1205–1211

47.62 S. Quinlan, O. Khatib: Elastic bands: Connecting path planning and control, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1993) pp. 802–807

47.63 O. Brock, O. Khatib: Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2000) pp. 550–555

47.64 F. Lamiraux, D. Bonnafous, O. Lefebvre: Reactive path deformation for nonholonomic mobile robots, IEEE Trans. Robotics **20**(6), 967–977 (2004)

47.65 Y. Yang, O. Brock: Elastic roadmaps – Motion generation for autonomous mobile manipulation, Auton. Robots **28**(1), 113–130 (2010)

47.66 E. Yoshida, C. Esteves, I. Belousov, J.-P. Laumond, T. Sakaguchi, K. Yokoi: Planning 3-d collision-free dynamic robotic motion through iterative reshaping, IEEE Trans. Robotics **24**, 1186–1198 (2008)

47.67 H. Kurniawati, T. Fraichard: From path to trajectory deformation, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2007)

47.68 O. Brock, O. Khatib: High-speed navigation using the global dynamic window approach, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1999) pp. 341–346

47.69 I. Ulrich, J. Borenstein: VFH*: Local obstacle avoidance with look-ahead verification, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2000) pp. 2505–2511

47.70 J. Minguez, L. Montano: Sensor-based motion robot motion generation in unknown, dynamic and troublesome scenarios, Robotics Auton. Syst. **52**(4), 290–311 (2005)

47.71 C. Stachniss, W. Burgard: An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2002) pp. 508–513

47.72 R. Philipsen, R. Siegwart: Smooth and efficient obstacle avoidance for a tour guide robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2003)

47.73 L. Montesano, J. Minguez, L. Montano: Lessons learned in integration for sensor-based robot navigation systems, Int. J. Adv. Robotics Syst. **3**(1), 85–91 (2006)

47.74 F. Lu, E. Milios: Robot pose estimation in unknown environments by matching 2-D range scans, Intell. Robotic Syst. **18**, 249–275 (1997)

47.75 J. Minguez, L. Montesano, F. Lamiraux: Metric-based iterative closest point scan matching for sensor displacement estimation, IEEE Trans. Robotics **22**(5), 1047–1054 (2006)

47.76 A. Stenz: The focussed d^* Algorithm for real-time replanning, Int. Jt. Conf. Artif. Intell. (IJCAI) (1995) pp. 1652–1659

47.77 S. Koenig, M. Likhachev: Improved fast replanning for robot navigation in unknown terrain, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2002)

47.78 J.-P. Laumond: Robot motion planning and control. In: *Lecture Notes in Control and Information Science*, ed. by J.P. Laumond (Springer, New York 1998)

47.79 H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, S. Thrun: *Principles of Robot Motion* (MIT Press, Cambridge 2005)

47.80 S.M. LaValle: *Planning Algorithms* (Cambridge Univ. Press, Cambridge 2006)