BESTELLUNG Semesterapparat DigiSem

MTP-000000289 **Bestelldatum: 2014-12-02 10:40:52**

Benutzernummer	
Name	Leibold, Marion
Straße	Technische Universität München
Postleitzahl	80797
Ort/Stadt	München
E-Mail-Adresse	marion.sobotka@tum.de

Unter Anerkennung des Urheberrechtsgesetzes wird bestellt:

ISBN	1-84628-641-7
Haupt-Titel	Robotics
Autor	Siciliano, Bruno
Titel	Path Planning
Band/Heft	()
Jahrgang	2010
Seiten	523-559

Signatur 0303/FER 980f 2011 L 291

Vermerk der Bibliothek

- o Jahrgang nicht vorhanden
- o verliehen
- o nicht am Standort
- o beim Buchbinder
- o vermißt
- o Sonstiges

12

Motion Planning

The trajectory planning methods presented in Chaps. 4 and 11, respectively for manipulators and mobile robots, operate under the simplifying assumption that the workspace is empty. In the presence of obstacles, it is necessary to plan motions that enable the robot to execute the assigned task without colliding with them. This problem, referred to as *motion planning*, is the subject of this chapter. After defining a canonical version of the problem, the concept of *configuration space* is introduced in order to achieve an efficient formulation. A selection of representative planning techniques is then presented. The method based on the notion of *retraction* characterizes the connectivity of the free configuration space using a *roadmap*, i.e., a set of collision-free paths, while the *cell decomposition* method identifies a network of *channels* with the same property. The *PRM* and *bidirectional RRT* techniques are *probabilistic* in nature and rely on the randomized sampling of the configuration space and the memorization of those samples that do not cause a collision between the robot and the obstacles. The *artificial potential* method is also described as a heuristic approach particularly suited to *on-line* planning problems, where the geometry of the workspace obstacles is unknown in advance. The chapter ends with a discussion of the application of the presented planning methods to the *robot manipulator* case.

12.1 The Canonical Problem

Robotic systems are expected to perform tasks in a workspace that is often populated by physical objects, which represent an obstacle to their motion. For example, a manipulator working in a robotized cell must avoid collision with its static structures, as well as with other moving objects that may access it, such as other manipulators. Similarly, a mobile robot carrying baggage in an airport has to navigate among obstacles that may be fixed (fittings, conveyor belts, construction elements) or mobile (passengers, workers). Planning a motion amounts then to deciding which path the robot must follow in order

to execute a transfer task from an initial to a final posture without colliding with the obstacles. Clearly, one would like to endow the robot with the capability of *autonomously* planning its motion, starting from a high-level description of the task provided by the user and a geometric characterization of the workspace, either made available entirely in advance (*off-line* planning) or gathered by the robot itself during the motion by means of on-board sensors (*on-line* planning).

However, developing automatic methods for motion planning is a very difficult endeavour. In fact, the spatial reasoning that humans instinctively use to move safely among obstacles has proven hard to replicate and codify in an algorithm that can be executed by a robot. To this date, motion planning is still an active topic of research, with contributions coming from different areas such as algorithm theory, computational geometry and automatic control.

To address the study of methods for motion planning, it is convenient to introduce a version of the problem that highlights its fundamental issues. The *canonical problem* of motion planning is then formulated as follows.

Consider a robot $\mathcal{B}$, which may consist of a single rigid body (mobile robot) or of a kinematic chain whose base is either fixed (standard manipulator) or mobile (mobile robot with trailers or mobile manipulator). The robot moves in a Euclidean space $\mathcal{W} = \mathbb{R}^N$, with $N = 2$ or 3, called *workspace*. Let $\mathcal{O}_1, \ldots, \mathcal{O}_p$ be the *obstacles*, i.e., fixed rigid objects in $\mathcal{W}$. It is assumed that both the geometry of $\mathcal{B}$, $\mathcal{O}_1, \ldots, \mathcal{O}_p$ and the pose of $\mathcal{O}_1, \ldots, \mathcal{O}_p$ in $\mathcal{W}$ are known. Moreover, it is supposed that $\mathcal{B}$ is *free-flying*, that is, the robot is not subject to any kinematic constraint. The motion planning problem is the following: given an initial and a final posture of $\mathcal{B}$ in $\mathcal{W}$, find if exists a *path*, i.e., a continuous sequence of postures, that drives the robot between the two postures while avoiding collisions (including contacts) between $\mathcal{B}$ and the obstacles $\mathcal{O}_1, \ldots, \mathcal{O}_p$; report a failure if such a path does not exist.

In the particular case in which the robot is a single body moving in $\mathbb{R}^2$, the canonical motion planning problem is also known as the *piano movers' problem*, as it captures the difficulties faced by movers when manoeuvring a piano (without lifting it) among obstacles. The *generalized movers' problem* is the canonical problem for a single-body robot moving in $\mathbb{R}^3$.

Clearly, some of the hypotheses of the canonical problem may not be satisfied in applications. For example, the assumption that the robot is the only object in motion in the workspace rules out the relevant case of moving obstacles (e.g., other robots). Advance knowledge of obstacle geometry and placement is another strong assumption: especially in *unstructured* environments, which are not purposely designed to accommodate robots, the robot itself is typically in charge of detecting obstacles by means of its sensors, and the planning problem must therefore be solved on-line during the motion. Moreover, as shown in Chap. 11, the free-flying robot hypothesis does not hold in nonholonomic mechanical systems, which cannot move along arbitrary paths in the workspace. Finally, manipulation and assembly problems are excluded from the canonical formulation since they invariably involve contacts between rigid

bodies. As a matter of fact, all the above assumptions are introduced in order to reduce motion planning to the purely geometrical — but still quite difficult — problem of generating a collision-free path. However, many methods that successfully solve this simplified version of the problem lend themselves to an extension to more general versions.

The notion of configuration space is essential to obtain a more convenient formulation of the canonical motion planning problem, as well as to envisage approaches to its solution.

12.2 Configuration Space

A very effective scheme for motion planning is obtained by representing the robot as a mobile point in an appropriate space, where the images of the workspace obstacles are also reported. To this end, it is natural to refer to the generalized coordinates of the mechanical system, whose value identifies the *configuration* of the robot (see Sect. B.4). This associates to each posture of the latter a point in the *configuration space* $\mathcal{C}$, i.e., the set of all the configurations that the robot can assume.

Generalized coordinates of robots are essentially of two types. Cartesian coordinates are used to describe the position of selected points on the links of the kinematic chain and take value in Euclidean spaces. Angular coordinates are used to represent the orientations of the bodies; independently from the adopted representation (rotation matrices, Euler angles, quaternions), they take values in $SO(m)$ $(m = 2, 3)$, the *special orthonormal group* of real $(m \times m)$ matrices with orthonormal columns and determinant equal to 1 (see Sect. 2.2). It is well known that a minimal parameterization of $SO(m)$ requires $m(m-1)/2$ parameters. The configuration space of a robot is then obtained in general as a Cartesian product of these spaces.

Some examples of configuration spaces are presented below:

- The configuration of a polygonal mobile robot in $\mathcal{W} = \mathbb{R}^2$ is described by the position of a representative point on the body (e.g., a vertex) and by the orientation of the polygon, both expressed with respect to a fixed reference frame. The configuration space $\mathcal{C}$ is then $\mathbb{R}^2 \times SO(2)$, whose dimension is $n = 3$.
- For a polyhedral mobile robot in $\mathcal{W} = \mathbb{R}^3$, the configuration space $\mathcal{C}$ is $\mathbb{R}^3 \times SO(3)$, whose dimension is $n = 6$.
- For a fixed-base planar manipulator with n revolute joints, the configuration space is a subset of $(\mathbb{R}^2 \times SO(2))^n$. The dimension of $\mathcal{C}$ equals the dimension of $(\mathbb{R}^2 \times SO(2))^n$ minus the number of constraints due to the presence of the joints, i.e., $3n-2n = n$. In fact, in a planar kinematic chain, each joint imposes two holonomic constraints on the following body.
- For a fixed-base spatial manipulator with n revolute joints, the configuration space is a subset of $(\mathbb{R}^3 \times SO(3))^n$. Since in this case each joint imposes five constraints on the following body, the dimension of $\mathcal{C}$ is $6n - 5n = n$.

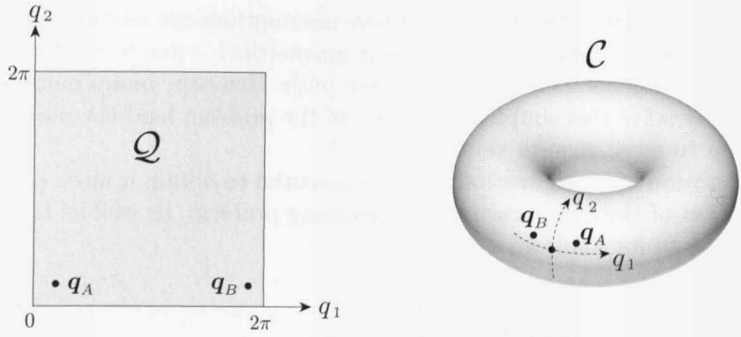

Fig. 12.1. The configuration space of a 2R manipulator; *left*: a locally valid representation as a subset of $\mathbb{R}^2$, *right*: a topologically correct representation as a two-dimensional torus

- For a unicycle-like vehicle with a trailer in $\mathbb{R}^2$, the configuration space is a subset of $(\mathbb{R}^2 \times SO(2)) \times (\mathbb{R}^2 \times SO(2))$. If the trailer is connected to the unicycle by a revolute joint, the configuration of the robot can be described by the position and orientation of the unicycle and the orientation of the trailer. The dimension of $\mathcal{C}$ is therefore $n = 4$.

If n is the dimension of $\mathcal{C}$, a configuration in $\mathcal{C}$ can be described by a vector $\boldsymbol{q} \in \mathbb{R}^n$. However, this description is only valid locally: the geometric structure of the configuration space $\mathcal{C}$ is in general more complex than that of a Euclidean space, as shown in the following example.

Example 12.1

Consider the planar manipulator with two revolute joints (2R manipulator) of Fig. 2.14. The configuration space has dimension 2, and may be locally represented by $\mathbb{R}^2$, or more precisely by its subset

$$\mathcal{Q} = \{\boldsymbol{q} = (q_1, q_2) : q_1 \in [0, 2\pi), q_2 \in [0, 2\pi)\}.$$

This guarantees that the representation is injective, i.e., that a single value of $\boldsymbol{q}$ exists for each manipulator posture. However, this representation is not topologically correct: for example, the configurations denoted as $\boldsymbol{q}_A$ and $\boldsymbol{q}_B$ in Fig. 12.1, left, which correspond to manipulator postures that are 'close' in the workspace $\mathcal{W}$, appear to be 'far' in $\mathcal{Q}$. To take this into account, one should 'fold' the square $\mathcal{Q}$ onto itself (so as to make opposite sides meet) in sequence along its two axes. This procedure generates a ring, properly called *torus*, which can be visualized as a two-dimensional surface immersed in $\mathbb{R}^3$ (Fig. 12.1, right). The correct expression of this space is $SO(2) \times SO(2)$.

When the configuration of a robot (either articulated or mobile) includes angular generalized coordinates, its configuration space is properly described as an n-dimensional *manifold*, i.e., a space in which a neighbourhood of a point can be put in correspondence with $\mathbb{R}^n$ through a continuous bijective function whose inverse is also continuous (a *homeomorphism*).

12.2.1 Distance

Having discussed the nature of the robot configuration space, it is useful to define a distance function in $\mathcal{C}$. In fact, the planning methods that will be discussed in the following make use of this notion.

Given a configuration $\boldsymbol{q}$, let $\mathcal{B}(\boldsymbol{q})$ be the subset of the workspace $\mathcal{W}$ occupied by the robot $\mathcal{B}$, and $p(\boldsymbol{q})$ be the position in $\mathcal{W}$ of a point p on $\mathcal{B}$. Intuition suggests that the distance between two configurations $\boldsymbol{q}_A$ and $\boldsymbol{q}_B$ should go to zero when the two regions $\mathcal{B}(\boldsymbol{q}_A)$ and $\mathcal{B}(\boldsymbol{q}_B)$ tend to coincide. A definition that satisfies this property is

$$d_1(\boldsymbol{q}_A, \boldsymbol{q}_B) = \max_{p \in \mathcal{B}} \|p(\boldsymbol{q}_A) - p(\boldsymbol{q}_B)\|, \tag{12.1}$$

where $\|\cdot\|$ denotes the Euclidean distance in $\mathcal{W} = \mathbb{R}^N$. In other words, the distance between two configurations in $\mathcal{C}$ is the maximum displacement in $\mathcal{W}$ they induce on a point, as the point moves all over the robot.

However, the use of function d_1 is cumbersome, because it requires characterizing the volume occupied by the robot in the two configurations and the computation of the maximum distance in $\mathcal{W}$ between corresponding points. For algorithmic purposes, the simple Euclidean norm is often chosen as a configuration space distance:

$$d_2(\boldsymbol{q}_A, \boldsymbol{q}_B) = \|\boldsymbol{q}_A - \boldsymbol{q}_B\|. \tag{12.2}$$

Nevertheless, one must keep in mind that this definition is appropriate only when $\mathcal{C}$ is a Euclidean space. Going back to Example 12.1, it is easy to realize that, unlike $d_1(\boldsymbol{q}_A, \boldsymbol{q}_B)$, the Euclidean norm $d_2(\boldsymbol{q}_A, \boldsymbol{q}_B)$ does not represent correctly the distance on the torus. A possible solution is to modify the definition of d_2 by suitably computing differences of angular generalized coordinates (see Problem 12.2).

12.2.2 Obstacles

In order to characterize paths that represent a solution to the canonical motion planning problem — those that avoid collisions between the robot and the workspace obstacles — it is necessary to build the 'images' of the obstacles in the configuration space of the robot.

In the following, it is assumed that the obstacles are closed (i.e., they contain their boundaries) but not necessarily limited subsets of $\mathcal{W}$. Given an

obstacle $\mathcal{O}_i$ $(i = 1, \ldots, p)$ in $\mathcal{W}$, its image in configuration space $\mathcal{C}$ is called *$\mathcal{C}$-obstacle* and is defined as

$$\mathcal{CO}_i = \{\boldsymbol{q} \in \mathcal{C} : \mathcal{B}(\boldsymbol{q}) \cap \mathcal{O}_i \neq \emptyset\}. \tag{12.3}$$

In other words, $\mathcal{CO}_i$ is the subset of configurations that cause a collision (including simple contacts) between the robot $\mathcal{B}$ and the obstacle $\mathcal{O}_i$ in the workspace. The union of all $\mathcal{C}$-obstacles

$$\mathcal{CO} = \bigcup_{i=1}^{p} \mathcal{CO}_i \tag{12.4}$$

defines the *$\mathcal{C}$-obstacle region*, while its complement

$$\mathcal{C}_{\text{free}} = \mathcal{C} - \mathcal{CO} = \{\boldsymbol{q} \in \mathcal{C} : \mathcal{B}(\boldsymbol{q}) \cap \left(\bigcup_{i=1}^{p} \mathcal{O}_i\right) = \emptyset\} \tag{12.5}$$

is the *free configuration space*, that is, the subset of robot configurations that do not cause collision with the obstacles. A path in configuration space is called *free* if it is entirely contained in $\mathcal{C}_{\text{free}}$.

Although $\mathcal{C}$ in itself is a connected space — given two arbitrary configuration there exists a path that joins them — the free configuration space $\mathcal{C}_{\text{free}}$ may not be connected as a consequence of occlusions due to $\mathcal{C}$-obstacles. Note also that the assumption of free-flying robot in the canonical problem means that the robot can follow any path in the free configuration space $\mathcal{C}_{\text{free}}$.

It is now possible to give a more compact formulation of the canonical motion planning problem. Assume that the initial and final posture of the robot $\mathcal{B}$ in $\mathcal{W}$ are mapped to the corresponding configurations in $\mathcal{C}$, respectively called *start* configuration $\boldsymbol{q}_s$ and *goal* configuration $\boldsymbol{q}_g$. Planning a collision-free motion for the robot means then generating a safe path between $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ if they belong to the same connected component of $\mathcal{C}_{\text{free}}$, and reporting a failure otherwise.

12.2.3 Examples of Obstacles

In the following, the $\mathcal{C}$-obstacle generation procedure is presented in some representative cases. For the sake of simplicity, it is assumed that obstacles in $\mathcal{W}$ are either polygonal or polyhedral.

Example 12.2

Consider the case of a point robot $\mathcal{B}$. In this case, the configuration of the robot is described by the coordinates of point $\mathcal{B}$ in the workspace $\mathcal{W} = \mathbb{R}^N$ and the configuration space $\mathcal{C}$ is a copy of $\mathcal{W}$. Similarly, the $\mathcal{C}$-obstacles are copies of the obstacles in $\mathcal{W}$.

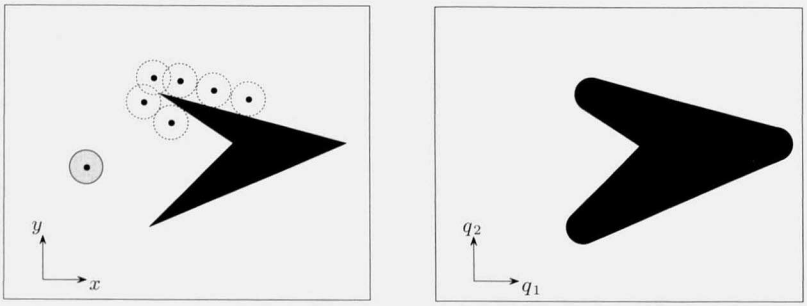

Fig. 12.2. $\mathcal{C}$-obstacles for a circular robot in $\mathbb{R}^2$; *left*: the robot $\mathcal{B}$, an obstacle $\mathcal{O}_i$ and the growing procedure for building $\mathcal{C}$-obstacles, *right*: the configuration space $\mathcal{C}$ and the $\mathcal{C}$-obstacle $\mathcal{CO}_i$

Example 12.3

If the robot is a sphere[1] in $\mathcal{W} = \mathbb{R}^N$, its configuration can be described by the Cartesian coordinates of a representative point, e.g., its centre — note that the orientation of the sphere is irrelevant for collision checking. Therefore, as in the previous example, the configuration space $\mathcal{C}$ is a copy of the workspace $\mathcal{W}$. However, the $\mathcal{C}$-obstacles are no longer simple copies of the obstacles in $\mathcal{W}$, and they must be built through a *growing* procedure. In particular, the boundary of the $\mathcal{C}$-obstacle $\mathcal{CO}_i$ is the locus of configurations that put the robot is in contact with the obstacle $\mathcal{O}_i$, and it can be obtained as the surface described by the representative point as the robot slides on the boundary of $\mathcal{O}_i$. As a consequence, to build $\mathcal{CO}_i$ it is sufficient to grow $\mathcal{O}_i$ isotropically by the radius of the robot. This procedure is shown in Fig. 12.2 for the case $N = 2$ (circular robot in $\mathbb{R}^2$); in this case, each $\mathcal{C}$-obstacle is a *generalized polygon*, i.e., a planar region whose boundary consists of line segments and/or circular arcs. If the representative point of the robot is different from the centre of the sphere, the growing procedure is not isotropic.

Example 12.4

Consider now the case of a polyhedral robot that is free to translate (with a fixed orientation) in $\mathbb{R}^N$. Its configuration can be described by the Cartesian coordinates of a representative point, for example a vertex of the polyhedron. Therefore, the configuration space $\mathcal{C}$ is again a copy of $\mathbb{R}^N$. Again, a growing procedure must be applied to the workspace obstacles to obtain their image in the configuration space. In particular, the boundary of the $\mathcal{C}$-obstacle $\mathcal{CO}_i$ is the surface described by the representative point of the robot when the robot $\mathcal{B}$ slides at a fixed orientation on the boundary of $\mathcal{O}_i$. Figure 12.3 shows this procedure for the case $N = 2$ (polygonal robot in $\mathbb{R}^2$). The resulting shape of $\mathcal{CO}_i$ depends on the position of the representative point on the robot, but in any case the $\mathcal{C}$-obstacle is itself a polyhedron. $\mathcal{CO}_i$ has in general a larger number of vertices than $\mathcal{O}_i$, and is a convex polyhedron provided that $\mathcal{B}$ and $\mathcal{O}_i$ are convex. Note also that, although the result of the growing

[1] For simplicity, the term *sphere* will be used in Euclidean spaces of arbitrary dimension n in place of *n–sphere*.

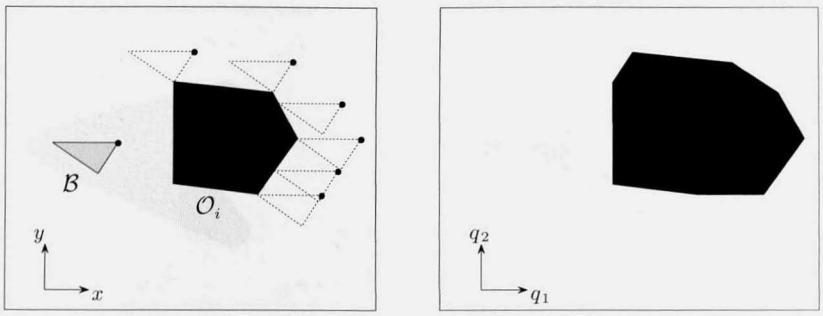

Fig. 12.3. $\mathcal{C}$-obstacles for a polygonal robot translating in $\mathbb{R}^2$; *left*: the robot $\mathcal{B}$, an obstacle $\mathcal{O}_i$ and the growing procedure for building $\mathcal{C}$-obstacles, *right*: the configuration space $\mathcal{C}$ and the $\mathcal{C}$-obstacle $\mathcal{CO}_i$

procedure — and thus the shape of the $\mathcal{C}$-obstacles — depends on the choice of the representative point on the robot, all the obtained planning problems in configuration space are equivalent. In particular, the existence of a solution for any of them implied the existence of a solution for all the others. Moreover, to each path in configuration space that solves one of these problems corresponds a (different) free path in any of the others, to which the same motion of the robot in the workspace is associated.

Example 12.5

For a polyhedral robot that can translate and rotate in $\mathbb{R}^N$, the dimension of the configuration space is increased with respect to the previous example, because it is also necessary to describe the orientation DOFs. For example, consider the case of a polygon that can translate and rotate in $\mathbb{R}^2$. The configuration of the robot can be characterized by the Cartesian coordinates of a representative point (for example, a vertex of the polygon) and an angular coordinate θ representing the orientation of the polygon with respect to a fixed reference frame. The configuration space $\mathcal{C}$ is then $\mathbb{R}^2 \times SO(2)$, which can be locally represented by $\mathbb{R}^3$. To build the image in $\mathcal{C}$ of an obstacle $\mathcal{O}_i$, one should in principle repeat the procedure illustrated in Fig. 12.3 for each possible value of the robot orientation θ. The $\mathcal{C}$-obstacle $\mathcal{CO}_i$ is the volume generated by 'stacking' (in the direction of the θ axis) all the constant-orientation slices thus obtained.

Example 12.6

For a robot manipulator $\mathcal{B}$ made by rigid links $\mathcal{B}_1, \ldots, \mathcal{B}_n$ connected by joints, there exist in principle two kinds of $\mathcal{C}$-obstacles: those that represent the collision between a body $\mathcal{B}_i$ and an obstacle $\mathcal{O}_j$, and those accounting for *self-collisions*, i.e., interference between two links $\mathcal{B}_i$ and $\mathcal{B}_j$ of the kinematic chain. Even considering for simplicity only the first type, the procedure for building $\mathcal{C}$-obstacles is much more complicated than in the previous examples. In fact, to obtain the boundary of the $\mathcal{C}$-obstacle $\mathcal{CO}_i$, it is necessary to identify through appropriate inverse kinematics computations all the configurations that bring one or more links of the manipulator $\mathcal{B}$ in contact with $\mathcal{O}_i$. Figure 12.4 shows the result of the $\mathcal{C}$-obstacle building

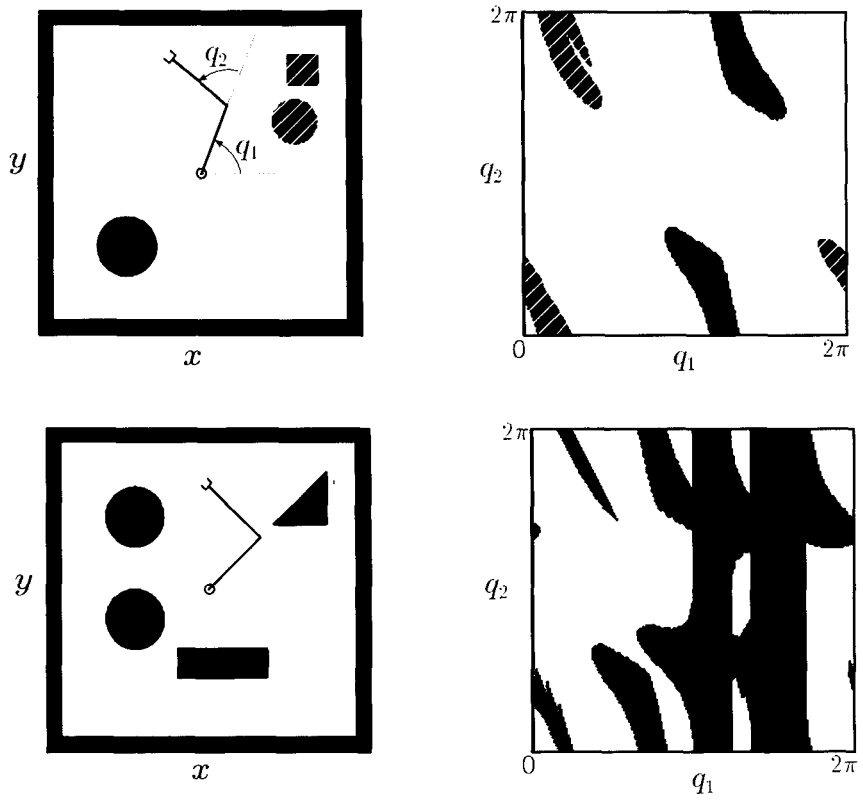

Fig. 12.4. $\mathcal{C}$-obstacles for a wire-frame 2R manipulator in two different cases; *left*: the robot and the obstacles in $\mathcal{W} = \mathbb{R}^2$, *right*: the configuration space $\mathcal{C}$ and the $\mathcal{C}$-obstacle region $\mathcal{CO}$

procedure for a wire-frame 2R manipulator in two different cases (self-collisions are not considered); note how, in spite of the simple shape of the obstacles in $\mathcal{W}$, the profile of the $\mathcal{C}$-obstacles is quite complicated. For simplicity, the configuration space has been represented as a subset (a square) of $\mathbb{R}^2$; however, to correctly visualize the $\mathcal{C}$-obstacles, one should keep in mind that the correct representation of $\mathcal{C}$ is a two-dimensional torus, so that the upper/lower and left/right sides of the square are actually coincident. Note also that in the first case (top of Fig. 12.4) the images of the two obstacles in the upper right corner of $\mathcal{W}$ merge in a single $\mathcal{C}$-obstacle, and the free configuration space $\mathcal{C}_{\text{free}}$ consists of a single connected component. In the second case (bottom of Fig. 12.4) $\mathcal{C}_{\text{free}}$ is instead partitioned into three disjoint connected components.

Whatever the nature of the robot, an *algebraic* model of the workspace obstacles is needed (e.g., derived from a CAD model of the workspace) to compute exactly the $\mathcal{C}$-obstacle region $\mathcal{CO}$. However, except for the most elementary cases, the procedures for generating $\mathcal{CO}$ are extremely complex. As a consequence, it is often convenient to resort to approximate representations of $\mathcal{CO}$. A simple (although computationally intensive) way to build such a representation is to extract configuration samples from $\mathcal{C}$ using a regular grid, compute the corresponding volume occupied by the robot via direct kinematics, and finally identify through *collision checking*[2] those samples that bring the robot to collide with obstacles. These samples can be considered as a discrete representation of $\mathcal{CO}$, whose accuracy can be improved at will by increasing the resolution of the grid in $\mathcal{C}$.

Some of the motion planning methods that will be presented in this chapter do not require an explicit computation of the $\mathcal{C}$-obstacle region. In particular, this is true for the probabilistic planners described in Sect. 12.5, and for the technique based on artificial potential fields and control points discussed in Sect. 12.7.

12.3 Planning via Retraction

The basic idea of motion planning via retraction is to represent the free configuration space by means of a *roadmap* $\mathcal{R} \subset \mathcal{C}_{\text{free}}$, i.e., a network of paths that describe adequately the connectivity of $\mathcal{C}_{\text{free}}$. The solution of a particular instance of a motion planning problem is then obtained by connecting (*retracting*) to the roadmap the start configuration $\boldsymbol{q}_s$ and the goal configuration $\boldsymbol{q}_g$, and finding a path on $\mathcal{R}$ between the two connection points. Depending on the type of roadmap, and on the retraction procedure, this general approach leads to different planning methods. One of these is described in the following, under the simplifying assumption that $\mathcal{C}_{\text{free}}$ is a limited subset of $\mathcal{C} = \mathbb{R}^2$ and is *polygonal*, i.e., its boundary is entirely made of line segments.[3] As the boundary of $\mathcal{C}_{\text{free}}$ coincides with the boundary of $\mathcal{CO}$, this assumption implies that the $\mathcal{C}$-obstacle region is itself a polygonal subset of $\mathcal{C}$.

For each configuration $\boldsymbol{q}$ in $\mathcal{C}_{\text{free}}$, let its *clearance* be defined as

$$\gamma(\boldsymbol{q}) = \min_{\boldsymbol{s} \in \partial\mathcal{C}_{\text{free}}} \|\boldsymbol{q} - \boldsymbol{s}\|, \tag{12.6}$$

where $\partial\mathcal{C}_{\text{free}}$ is the boundary of $\mathcal{C}_{\text{free}}$. The clearance $\gamma(\boldsymbol{q})$ represents the minimum Euclidean distance between the configuration $\boldsymbol{q}$ and the $\mathcal{C}$-obstacle re-

[2] Many algorithms based on computational geometry techniques are available (in the literature, but also implemented in software packages) to test collision in $\mathbb{R}^2$ o $\mathbb{R}^3$. The most efficient, such as *I-Collide* and *V-Collide*, use a hierarchical representation of geometric models of bodies, and can speed up collision checking by re-using the results of previous checks in spatially similar situations.

[3] According to the definition, a polygonal subset is not necessarily connected, and may contain 'holes'.

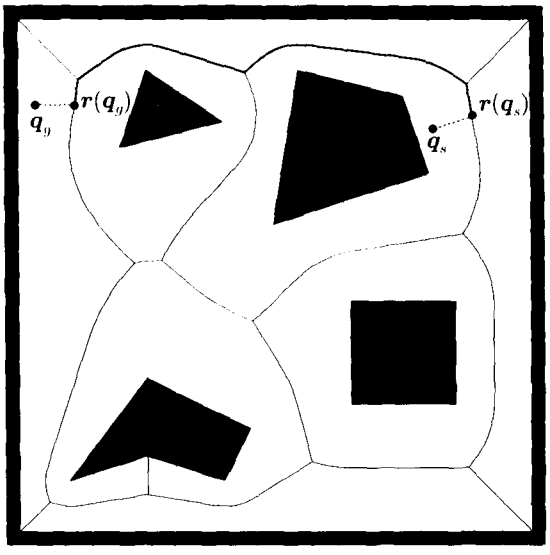

Fig. 12.5. An example of generalized Voronoi diagram and the solution of a particular instance of the planning problem, obtained by retracting q_s and q_g on the diagram. The solution path from q_s to q_g consists of the two *dashed* segments and the *thick* portion of the diagram joining them

gion. Moreover, consider the set of points on the boundary of $\mathcal{C}_{\text{free}}$ that are *neighbours* of $\boldsymbol{q}$:

$$N(\boldsymbol{q}) = \{\boldsymbol{s} \in \partial\mathcal{C}_{\text{free}} : \|\boldsymbol{q} - \boldsymbol{s}\| = \gamma(\boldsymbol{q})\}, \tag{12.7}$$

i.e., the points on $\partial\mathcal{C}_{\text{free}}$ that determine the value of the clearance for $\boldsymbol{q}$. With these definitions, the *generalized*[4] *Voronoi diagram* of $\mathcal{C}_{\text{free}}$ is the locus of its configurations having more than one neighbour:

$$\mathcal{V}(\mathcal{C}_{\text{free}}) = \{\boldsymbol{q} \in \mathcal{C}_{\text{free}} : \text{card}(N(\boldsymbol{q})) > 1\}, \tag{12.8}$$

in which $\text{card}(\cdot)$ denotes the cardinality of a set. Figure 12.5 shows an example of generalized Voronoi diagram for a polygonal free configuration space; note how the connectivity of $\mathcal{C}_{\text{free}}$ is well captured by the diagram.

It is easy to show that $\mathcal{V}(\mathcal{C}_{\text{free}})$ is made of elementary arcs that are either rectilinear — each made of contiguous configurations whose clearance is due to the same pair of edges or vertices — or parabolic — each made of contiguous configurations whose clearance is determined by the same pair edge-vertex. Therefore, one can build an analytical expression of $\mathcal{V}(\mathcal{C}_{\text{free}})$ starting from the pair of features (side/side, side/vertex, vertex/vertex) that determine the

[4] A proper *Voronoi diagram* is obtained in the particular case in which the $\mathcal{C}$-obstacles are isolated points.

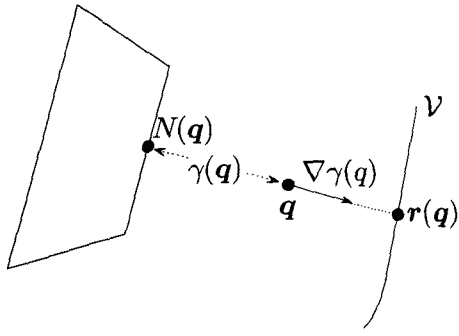

Fig. 12.6. The retraction procedure for connecting a generic configuration $\boldsymbol{q}$ in $\mathcal{C}_{\text{free}}$ to $\mathcal{V}(\mathcal{C}_{\text{free}})$

appearance of each arc. From an abstract point of view, $\mathcal{V}(\mathcal{C}_{\text{free}})$ can be considered as a *graph* having elementary arcs of the diagram as *arcs* and endpoints of arcs as *nodes*.

By construction, the generalized Voronoi diagram has the property of locally maximizing clearance, and is therefore a natural choice as a roadmap of $\mathcal{C}_{\text{free}}$ for planning motions characterized by a healthy safety margin with respect to the possibility of collisions. To use $\mathcal{V}(\mathcal{C}_{\text{free}})$ as a roadmap, a retraction procedure must be defined for connecting a generic configuration in $\mathcal{C}_{\text{free}}$ to the diagram. To this end, consider the geometric construction shown in Fig. 12.6. Since $\boldsymbol{q} \notin \mathcal{V}(\mathcal{C}_{\text{free}})$, it is $\text{card}\big(N(\boldsymbol{q})\big) = 1$, i.e., there exists a single point on the polygonal boundary of $\mathcal{CO}$ (either a vertex or a point on a side) that determines the value of the clearance $\gamma(\boldsymbol{q})$. The gradient $\nabla\gamma(\boldsymbol{q})$, which identifies the direction of steepest ascent for the clearance at the configuration $\boldsymbol{q}$, is directed as the half-line originating in $N(\boldsymbol{q})$ and passing through $\boldsymbol{q}$. The first intersection of this half-line with $\mathcal{V}(\mathcal{C}_{\text{free}})$ defines $\boldsymbol{r}(\boldsymbol{q})$, i.e., the connection point of $\boldsymbol{q}$ to the generalized Voronoi diagram. To guarantee that $\boldsymbol{r}(\cdot)$ is continuous, it is convenient to extend its domain of definition to all $\mathcal{C}_{\text{free}}$ by letting $\boldsymbol{r}(\boldsymbol{q}) = \boldsymbol{q}$ if $\boldsymbol{q} \in \mathcal{V}(\mathcal{C}_{\text{free}})$.

From a topological viewpoint, the function $\boldsymbol{r}(\cdot)$ defined above is actually an example of *retraction* of $\mathcal{C}_{\text{free}}$ on $\mathcal{V}(\mathcal{C}_{\text{free}})$, i.e., a continuous surjective map from $\mathcal{C}_{\text{free}}$ to $\mathcal{V}(\mathcal{C}_{\text{free}})$ such that its restriction to $\mathcal{V}(\mathcal{C}_{\text{free}})$ is the identity map. In view of its definition, $\boldsymbol{r}(\cdot)$ preserves the connectivity of $\mathcal{C}_{\text{free}}$, in the sense that $\boldsymbol{q}$ and $\boldsymbol{r}(\boldsymbol{q})$ — as well as the segment joining them — always lie in the same connected component of $\mathcal{C}_{\text{free}}$. This property is particularly important, because it is possible to show that, given a generic connectivity-preserving retraction ρ of $\mathcal{C}_{\text{free}}$ on a roadmap $\mathcal{R}$, there exists a free path between two configurations $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ if and only if there exists a path on $\mathcal{R}$ between $\rho(\boldsymbol{q}_s)$ and $\rho(\boldsymbol{q}_g)$. As a consequence, the problem of planning a path in $\mathcal{C}_{\text{free}}$ reduces to the problem of planning a path on its retraction $\mathcal{R} = \rho(\mathcal{C}_{\text{free}})$.

Given the start and goal configurations $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, the motion planning method via retraction goes through the following steps (see Fig. 12.5):

1. Build the generalized Voronoi diagram $\mathcal{V}(\mathcal{C}_{\text{free}})$.
2. Compute the retractions $\boldsymbol{r}(\boldsymbol{q}_s)$ and $\boldsymbol{r}(\boldsymbol{q}_g)$ on $\mathcal{V}(\mathcal{C}_{\text{free}})$.
3. Search $\mathcal{V}(\mathcal{C}_{\text{free}})$ for a sequence of consecutive arcs such that $\boldsymbol{r}(\boldsymbol{q}_s)$ belongs to the first and $\boldsymbol{r}(\boldsymbol{q}_g)$ to the last.
4. If the search is successful, replace the first arc of the sequence with its subarc originating in $\boldsymbol{r}(\boldsymbol{q}_s)$ and the last arc of the sequence with its subarc terminating in $\boldsymbol{r}(\boldsymbol{q}_g)$, and provide as output the path consisting of the line segment joining $\boldsymbol{q}_s$ to $\boldsymbol{r}(\boldsymbol{q}_s)$, the modified arc sequence, and the line segment joining $\boldsymbol{q}_g$ to $\boldsymbol{r}(\boldsymbol{q}_g)$; otherwise, report a failure.

If simplicity of implementation is desired, the graph search[5] required at Step 3 can be performed using basic strategies such as breadth-first or depth-first search. On the other hand, if one wishes to identify the minimum-length path (among those which can be produced by the method) between $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, each arc must be labelled with a cost equal to its actual length. The minimum-cost path can then be computed with an *informed* algorithm — i.e., an algorithm using a heuristic estimate of the minimum cost path from a generic node to the goal, in this case the Euclidean distance — such as $A^\star$.

Whatever the adopted search strategy, the above motion planning method via retraction of $\mathcal{C}_{\text{free}}$ on $\mathcal{V}(\mathcal{C}_{\text{free}})$ is *complete*, i.e., it is guaranteed to find a solution if one exists, and to report a failure otherwise. Its time complexity is a function of the number v of vertices of the polygonal region $\mathcal{C}_{\text{free}}$, and depends essentially on the construction of the generalized Voronoi diagram (Step 1), on the retraction of $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ (Step 2), and on the search on the diagram (Step 3). As for Step 1, the most efficient algorithms can build $\mathcal{V}(\mathcal{C}_{\text{free}})$ in time $O(v \log v)$. The retraction procedure requires $O(v)$, mainly to compute $N(\boldsymbol{q}_s)$ and $N(\boldsymbol{q}_g)$. Finally, since it is possible to prove that $\mathcal{V}(\mathcal{C}_{\text{free}})$ has $O(v)$ arcs, the complexity of breadth-first or depth-first search would be $O(v)$, whereas $A^\star$ would require $O(v \log v)$. Altogether, the time complexity of the motion planning method via retraction is $O(v \log v)$.

It should be noted that, once the generalized Voronoi diagram of $\mathcal{C}_{\text{free}}$ has been computed, it can be used again to solve quickly other instances (*queries*) of the same motion planning problem, i.e., to generate collision-free paths between different start and goal configurations in the same $\mathcal{C}_{\text{free}}$. For example, this is useful when a robot must repeatedly move between different postures in the same static workspace. The motion planning method based on retraction can then be considered *multiple-query*. It is also possible to extend the method to the case in which the $\mathcal{C}$-obstacles are generalized polygons.

[5] See Appendix E for a quick survey on graph search strategies and algorithm complexity.

12.4 Planning via Cell Decomposition

Assume that the free configuration space $\mathcal{C}_{\text{free}}$ can be decomposed in simply-shaped regions, called *cells*, with the following basic characteristics:

- Given two configurations belonging to the same cell, it is 'easy' to compute a collision-free path that joins them.
- Given two adjacent cells — i.e., two cells having in common a portion of their boundaries of non-zero measure — it is 'easy' to generate a collision-free path going from one cell to the other.

Starting from one such cell decomposition of $\mathcal{C}_{\text{free}}$, it is easy to build the associated *connectivity graph*. The nodes of this graph represent cells, while an arc between two nodes indicates that the two corresponding cells are adjacent. By searching the connectivity graph for a path from the cell containing the start configuration $\boldsymbol{q}_s$ to the cell containing the goal configuration $\boldsymbol{q}_g$, one obtains (if it exists) a sequence of adjacent cells, called *channel*, from which it is possible — in view of the above mentioned characteristics of cells — to extract a path that joins $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$ and is entirely contained in $\mathcal{C}_{\text{free}}$.

The general approach so far outlined generates different motion planning methods, essentially depending on the type of cells used for the decomposition. In the following, two algorithms are described, respectively based on exact and approximate cell decomposition of $\mathcal{C}_{\text{free}}$. As before, it is assumed that $\mathcal{C}_{\text{free}}$ is a limited polygonal subset of $\mathcal{C} = \mathbb{R}^2$.

12.4.1 Exact Decomposition

When an exact decomposition is used, the free configuration space is partitioned in a collection of cells whose union gives exactly $\mathcal{C}_{\text{free}}$. A typical choice for cells are convex polygons. In fact, convexity guarantees that the line segments joining two configurations belonging to the same cell lies entirely in the cell itself, and therefore in $\mathcal{C}_{\text{free}}$. Moreover, it is easy to travel safely between two adjacent cells by passing through the midpoint of the segment that constitutes their common boundary. A simple way to decompose $\mathcal{C}_{\text{free}}$ in a collection of convex polygons is to use the *sweep line* algorithm, which turns out to be useful in a number of computational geometry problems. In the present setting, the application of this algorithm proceeds as follows.

Choose a line that is not parallel to any side of the boundary of $\mathcal{C}_{\text{free}}$, and let it translate ('sweep') all over $\mathcal{C}_{\text{free}}$. Whenever the line passes through one of the vertices of $\mathcal{C}_{\text{free}}$, consider the two segments (*extensions*) that originate from the vertex, lie on the line and point in opposite directions, terminating at the first intersection with $\partial\mathcal{C}_{\text{free}}$. Every extension that lies in $\mathcal{C}_{\text{free}}$ (except for its endpoints) is part of the boundary of a cell; the rest of the boundary is made of (parts of) sides of $\partial\mathcal{C}_{\text{free}}$ and possibly other extensions. This procedure is illustrated in Fig. 12.7, where a vertical sweep line has been used; note that some of the vertices of $\mathcal{C}_{\text{free}}$ contribute to the decomposition with a single or no

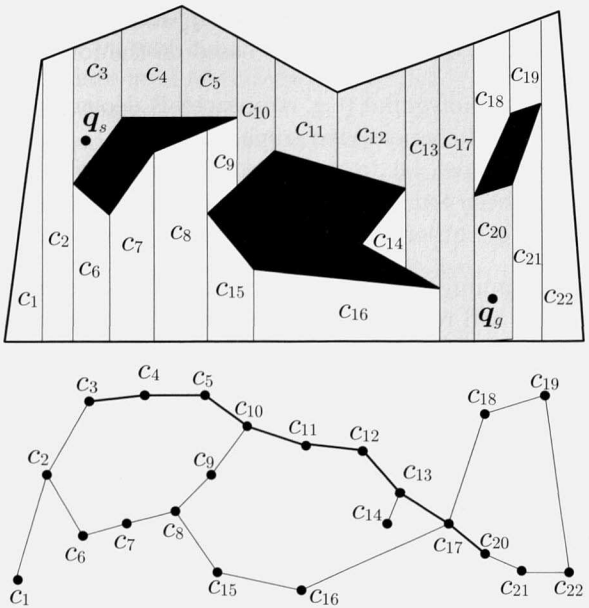

Fig. 12.7. An example of trapezoidal decomposition via the sweep line algorithm (*above*) and the associated connectivity graph (*below*). Also shown is the solution of a particular planning problem ($c_s = c_3$ and $c_g = c_{20}$), both as a channel in $\mathcal{C}_{\text{free}}$ and as a path on the graph

extension at all. The result is a special case of *convex polygonal decomposition*, that is called *trapezoidal* because its cells are trapezoids — triangular cells, if present, are regarded as degenerate trapezoids having one side of zero length.

After the decomposition of $\mathcal{C}_{\text{free}}$ has been computed, it is possible to build the associated *connectivity graph* C. This is the graph whose nodes are the cells of the decomposition, while an arc exists between two nodes if the corresponding cells are adjacent, i.e., the intersection of their boundary is a line segment of non-zero length; therefore, cells with side-vertex or vertex-vertex contacts are not adjacent. At this point, it is necessary to identify the cells c_s and c_g in the decomposition that respectively contain $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, the start and goal configurations for the considered planning problem. A graph search algorithm can then be used to find a *channel* from c_s to c_g, i.e., a path on C that joins the two corresponding nodes (see Fig. 12.7). From the channel, which is a sequence of adjacent cells, one must extract a path in $\mathcal{C}_{\text{free}}$ going from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$. Since the interior of the channel is contained in $\mathcal{C}_{\text{free}}$ and the cells are convex polygons, such extraction is straightforward. For example, one may identify the midpoints of the segments that represent the common boundaries between consecutive cells of the channel, and connect them through a broken line starting in $\boldsymbol{q}_s$ and ending in $\boldsymbol{q}_g$.

Wrapping up, given the two configurations $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, the motion planning algorithm via exact cell decomposition is based on the following steps:

1. Compute a convex polygonal (e.g., trapezoidal) decomposition of $\mathcal{C}_{\text{free}}$.
2. Build the associated connectivity graph C.
3. Search C for a channel, i.e., a sequence of adjacent cells from c_s to c_g.
4. If a channel has been found, extract and provide as output a collision-free path from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$; otherwise, report a failure.

As in motion planning via retraction, using a non-informed graph search algorithm in Step 3 will result in a channel that is not optimal, in the sense that all paths from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$ that can be extracted from the channel may be longer than necessary. To compute efficient paths, the use of $A^\star$ is advisable as a search algorithm. To this end, one should build a modified connectivity graph C' having as nodes $\boldsymbol{q}_s$, $\boldsymbol{q}_g$ and all the midpoints of adjacency segments between cells, and arcs joining nodes belonging to the same cell (note that nodes on adjacency segments belong to two cells). Each arc is then labelled with a cost equal to the distance between the nodes connected by the arc. If the heuristic function is chosen as the distance between the current node and $\boldsymbol{q}_g$, the use of $A^\star$ will produce the shortest path in C', if a solution exists.

The motion planning method based on exact cell decomposition is complete. As for its time complexity, it depends mainly on the cell decomposition and on the connectivity graph search. Using the sweep line algorithm, the decomposition procedure (including the generation of the connectivity graph) has complexity $O(v \log v)$, where v is the number of vertices of $\mathcal{C}_{\text{free}}$. Moreover, it can be shown that the connectivity graph C has $O(v)$ arcs. Hence, regardless of the adopted search strategy, the motion planning method based on exact cell decomposition has complexity $O(v \log v)$.

Note the following facts:

- Any planner based on exact cell decomposition can be considered multiple-query. In fact, once computed, the connectivity graph associated with the decomposition can be used to solve different instances of the same motion planning problem.
- The connectivity graph represents a network of channels, each implicitly containing an *infinity* of paths that traverse the channel and are topologically equivalent, i.e., differ only for a continuous deformation. Therefore, cell decomposition provides as output a structure that is more flexible than the roadmap used by retraction-based methods. This may be useful to plan paths in the channel that are also admissible with respect to possible kinematic constraints, as well as to avoid unexpected obstacles during the actual execution of the motion.
- The solution paths produced by the planning method based on exact cell decomposition are broken lines. It is however possible to smooth the path using *curve fitting* techniques. In practice, one selects a sufficient number

of intermediate points (*via points*) on the path, among which it is necessary to include $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, and then interpolates them using functions with an appropriate level of differentiability (e.g., polynomial functions of sufficiently high degree).

- The above method can be extended to the case in which $\mathcal{C}_{\text{free}}$ is a limited polyhedral subset of $\mathcal{C} = \mathbb{R}^3$. In particular, the decomposition of $\mathcal{C}_{\text{free}}$ can obtained through the *sweep plane* algorithm, which produces polyhedral cells. The common boundary between adjacent cells consists of trapezoid of non-zero area. This boundary can be safely crossed, e.g., at the barycentre of the trapezoid.

Finally, it should be mentioned that there exist in the literature methods based on exact cell decomposition that can solve essentially any motion planning problem, regardless of the dimension of $\mathcal{C}$ and of the geometry of $\mathcal{C}_{\text{free}}$, which can also be non-polyhedral. However, the complexity of these planners is prohibitive, being exponential in the dimension of $\mathcal{C}$, and their importance is therefore mainly theoretical.

12.4.2 Approximate Decomposition

In approximate decompositions of $\mathcal{C}_{\text{free}}$, disjoint cells of predefined shape are used; for example, when $\mathcal{C} = \mathbb{R}^2$ one may choose square or rectangular cells. In general, the union of all cells will represent an approximation by defect of $\mathcal{C}_{\text{free}}$. To achieve a reasonable trade-off between the accuracy of the approximation and the efficiency of the decomposition procedure, a recursive algorithm is typically used, which starts with a coarse grid whose resolution is locally increased to adapt better to the geometry of $\mathcal{C}_{\text{free}}$. As in motion planning methods based on exact decomposition, the connectivity graph associated with the obtained approximate decomposition is searched for a channel, from which a solution path can be extracted.

In the following, a motion planning method based on approximate decomposition is described for the case in which $\mathcal{C}_{\text{free}}$ is a limited polygonal subset of $\mathcal{C} = \mathbb{R}^2$. Without loss of generality, it is assumed that the 'external' boundary of $\mathcal{C}_{\text{free}}$ is a square, and square cells are therefore used. The decomposition algorithm (Fig. 12.8) starts by dividing initially the square containing $\mathcal{C}_{\text{free}}$ into four cells, that are classified according to the categories below:

- *free* cells, whose interior has no intersection with the $\mathcal{C}$-obstacle region;
- *occupied* cells, entirely contained in the $\mathcal{C}$-obstacle region;
- *mixed* cells, that are neither free nor occupied.

At this point, one builds the connectivity graph C associated with the current level of decomposition: this is the graph having free and mixed cells as nodes, and arcs that join nodes representing adjacent cells. Once the nodes corresponding to the cells that contain $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ have been identified, C is searched for a path between them, e.g., using the $A^\star$ algorithm. If such a path

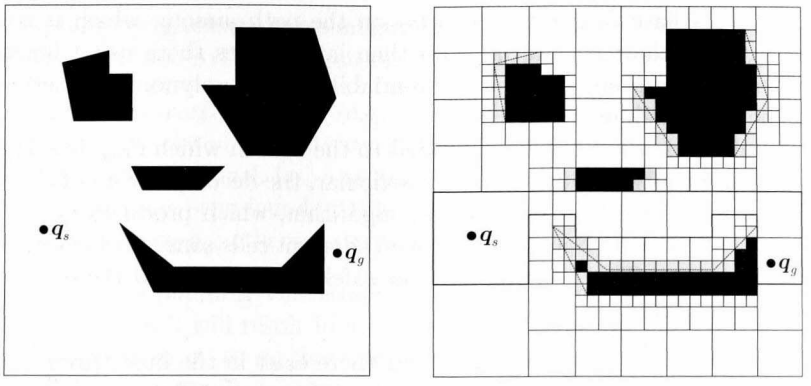

Fig. 12.8. An example of motion planning via approximate cell decomposition; *left*: the assigned problem, *right*: the solution as a free channel (*thick line*)

does not exist, a failure is reported. If the path exists, it consists of a sequence of cells that may be either all free (*free channel*) or not (*mixed channel*). In the first case, a solution to the motion planning problem has been found; in particular, a configuration space path can be easily extracted from the free channel as in the method based on exact cell decomposition. Instead, if the channel contains mixed cells, each of them is further divided into four cells, which are then classified as free, occupied or mixed. The algorithm proceeds by iterating these steps, until a free channel going from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$ has been found or a minimum admissible size has been reached for the cells. Figure 12.8 shows an example of application of this technique. Note that, at the resolution level where the solution has been found, free and occupied cells represent an approximation by defect of $\mathcal{C}_{\text{free}}$ and $\mathcal{CO}$, respectively. The missing part of $\mathcal{C}$ is occupied by mixed cells (in gray).

At each iteration, the search algorithm on the connectivity graph can find a free channel going from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$ only if such a channel exists on the approximation of $\mathcal{C}_{\text{free}}$ (i.e., on the free cells) at the current level of decomposition. This motion planning method is therefore *resolution complete*, in the sense that a sufficient reduction of the minimum admissible size for cells guarantees that a solution is found whenever one exists.

A comparison between Figs. 12.7 and 12.8 clearly shows that, unlike what happens in exact decomposition, the boundary of a cell in an approximate decomposition does not correspond in general to a change in the spatial constraints imposed by obstacles. One of the consequences of this fact is that the implementation of a motion planning method based on approximate decomposition is remarkably simpler, as it only requires a recursive division of cells followed by a collision check between the cells and the $\mathcal{C}$-obstacle region. In particular, the first can be realized using a data structure called *quadtree*. This is a tree in which any internal node (i.e., a node that is not a leaf) has exactly four child nodes. In the cell decomposition case, the tree is rooted at

$\mathcal{C}$, i.e., the whole configuration space. Lower level nodes represent cells that are free, occupied or mixed; only the latter have children.

Another difference with respect to the method based on exact decomposition is that an approximate decomposition is intrinsically associated with a particular instance of a planning problem, as the decomposition procedure itself is guided by the search for a free channel between start and goal.

The planning method based on approximate decomposition is conceptually applicable in configuration spaces of arbitrary dimension. For example, in $\mathbb{R}^3$ it is possible to use an *octree*, a tree in which any internal node has eight children. In $\mathbb{R}^n$, the corresponding data structure is a 2^n*-tree*. Since the maximum number of leaves of a 2^n-tree is 2^{np}, where p is the depth (number of levels) of the tree, the complexity of approximate cell decomposition — and thus of the associated motion planning method — is exponential in the dimension of $\mathcal{C}$ and in the maximal resolution of the decomposition. As a consequence, this technique is effective in practice only in configuration spaces of low dimension (typically, not larger than 4).

12.5 Probabilistic Planning

Probabilistic planners represent a class of methods of remarkable efficiency, especially in problems involving high-dimensional configuration spaces. They belong to the general family of *sampling-based* methods, whose basic idea consists of determining a finite set of collision-free configurations that adequately represent the connectivity of $\mathcal{C}_{\text{free}}$, and using these configurations to build a roadmap that can be employed for solving motion planning problems. This is realized by choosing at each iteration a sample configuration and checking if it entails a collision between the robot and the workspace obstacles. If the answer is affirmative, the sample is discarded. A configuration that does not cause a collision is instead added to the current roadmap and connected if possible to other already stored configurations.

The above strategy is quite general and may lead to different planning methods depending on the specific design choices, and mainly on the criterion for selecting the samples in $\mathcal{C}$ to be checked for collision. One may proceed in a *deterministic* fashion, choosing the samples by means of a regular grid that is applied to $\mathcal{C}$. However, it is preferable to use a *randomized* approach, in which the sample configurations are chosen according to some probability distribution. In the following, two planners of this type are described.

12.5.1 PRM Method

The basic iteration of the PRM (*Probabilistic Roadmap*) method begins by generating a random sample $\boldsymbol{q}_{\text{rand}}$ of the configuration space using a uniform probability distribution in $\mathcal{C}$. Then, $\boldsymbol{q}_{\text{rand}}$ is tested for collision, by using kinematic and geometric relationships to compute the corresponding posture of

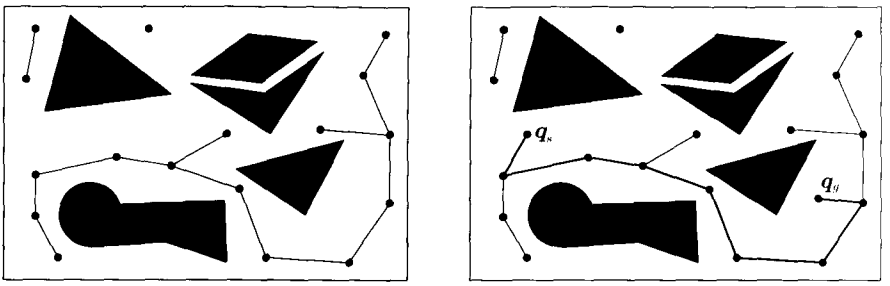

Fig. 12.9. A PRM in a two-dimensional configuration space (*left*) and its use for solving a particular planning problem (*right*)

the robot and invoking an algorithm that can detect collisions (including contacts) between the latter and the obstacles. If $\boldsymbol{q}_{\text{rand}}$ does not cause collisions. it is added to the roadmap and connected (if possible) through free *local paths* to sufficiently 'near' configurations already in the roadmap. Usually, 'nearness' is defined on the basis of the Euclidean distance in $\mathcal{C}$, but it is possible to use different distance notions; for example, as mentioned in Sect. 12.2.1. one may use a configuration space distance notion induced by a distance in the workspace. The generation of a free local path between $\boldsymbol{q}_{\text{rand}}$ and a near configuration $\boldsymbol{q}_{\text{near}}$ is delegated to a procedure known as *local planner*. A common choice is to throw a rectilinear path in $\mathcal{C}$ between $\boldsymbol{q}_{\text{rand}}$ and $\boldsymbol{q}_{\text{near}}$ and test it for collision, for example by sampling the segment with sufficient resolution and checking the single samples for collision. If the local path causes a collision, it is discarded and no direct connection between $\boldsymbol{q}_{\text{rand}}$ and $\boldsymbol{q}_{\text{near}}$ appears in the roadmap.

The PRM incremental generation procedure stops when either a maximum number of iterations has been reached, or the number of connected components in the roadmap becomes smaller than a given threshold. At this point. one verifies whether it is possible to solve the assigned motion planning problem by connecting $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ to the *same* connected component of the PRM by free local paths. Figure 12.9 shows an example of PRM and the solution to a particular problem. Note the presence of multiple connected components of the PRM, one of which consists of a single configuration.

If a solution cannot be found, the PRM can be improved by performing more iterations of the basic procedure, or using special strategies aimed at reducing the number of its connected components. For example, a possible technique consists of trying to connect configurations that are close but belong to different components via more general (e.g., not rectilinear) local paths.

The main advantage of the PRM method is its remarkable speed in finding a solution to motion planning problems, provided that the roadmap has been sufficiently developed. In this respect, it should be noted that new instances of the same problem induce a potential enhancement of the PRM, which improves with usage both in terms of connectivity and of time efficiency. The

PRM method is therefore intrinsically multiple-query. In high-dimensional configuration spaces, the time needed by this method to compute a solution can be several orders of magnitude smaller than with the previously presented techniques. Another aspect of the method that is worth mentioning is the simplicity of implementation. In particular, note that the generation of $\mathcal{C}$-obstacles is completely eliminated.

The downside of the PRM method is that it is only *probabilistically complete*, i.e., the probability of finding a solution to the planning problem when one exists tends to 1 as the execution time tends to infinity. This means that, if no solution exists, the algorithm will run indefinitely. In practice, a maximum number of iterations is enforced so as to guarantee its termination.

A situation that is critical for the PRM method is the presence of *narrow passages* in $\mathcal{C}_{\text{free}}$, such as the one shown in the upper-right quadrant of the scene in Fig. 12.9. In fact, using a uniform distribution for generating $\boldsymbol{q}_{\text{rand}}$, the probability of placing a sample in a certain region of $\mathcal{C}_{\text{free}}$ is proportional to its volume. As a consequence, depending on its size, it may be very unlikely that a path crossing a narrow passage appears in the PRM within a reasonable time. To alleviate this problem, the method can be modified by using non-uniform probability distributions. For example, there exist strategies for generating $\boldsymbol{q}_{\text{rand}}$ that are biased towards those regions of $\mathcal{C}$ that contain fewer samples, and therefore are more likely to contain close obstacles and the associated narrow passages.

12.5.2 Bidirectional RRT Method

Single-query probabilistic methods are aimed at quickly solving a particular instance of a motion planning problem. Unlike multiple-query planners such as PRM, these techniques do not rely on the generation of a roadmap that represents exhaustively the connectivity of the free configuration space; in fact, they tend to explore only a subset of $\mathcal{C}_{\text{free}}$ that is relevant for solving the problem at hand. This results in a further reduction of the time needed to compute a solution.

An example of single-query probabilistic planner is the *bidirectional RRT* method, which makes use of a data structure called RRT (*Rapidly-exploring Random Tree*). The incremental expansion of an RRT, denoted by T in the following, relies on a simple randomized procedure to be repeated at each iteration (see Fig. 12.10). The first step is the generation of a random configuration $\boldsymbol{q}_{\text{rand}}$ according to a uniform probability distribution in $\mathcal{C}$ (as in the PRM method). Then, the configuration $\boldsymbol{q}_{\text{near}}$ in T that is closer to $\boldsymbol{q}_{\text{rand}}$ is found, and a new candidate configuration $\boldsymbol{q}_{\text{new}}$ is produced on the segment joining $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{rand}}$ at a predefined distance δ from $\boldsymbol{q}_{\text{near}}$. A collision check is then run to verify that both $\boldsymbol{q}_{\text{new}}$ and the segment going from $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{new}}$ belong to $\mathcal{C}_{\text{free}}$. If this is the case, T is *expanded* by incorporating $\boldsymbol{q}_{\text{new}}$ and the segment joining it to $\boldsymbol{q}_{\text{near}}$. Note that $\boldsymbol{q}_{\text{rand}}$ is not added to the tree, so

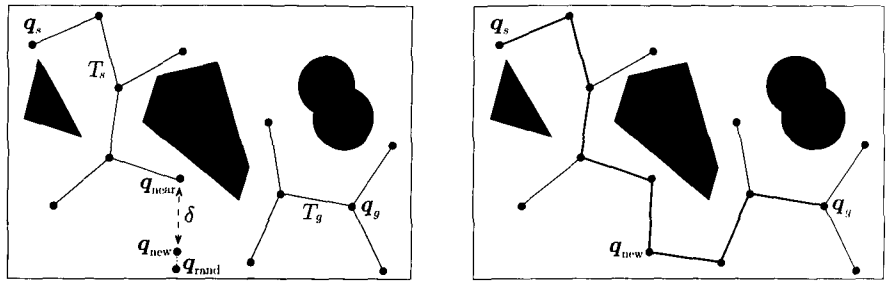

Fig. 12.10. The bidirectional RRT method in a two-dimensional configuration space *left*: the randomized mechanism for expanding a tree, *right*: the extension procedure for connecting the two trees

that it is not necessary to check whether it belongs to $\mathcal{C}_{\text{free}}$; its only function is to indicate a direction of expansion for T.

It is worth pointing out that the RRT expansion procedure, although quite simple, results in a very efficient 'exploration' of $\mathcal{C}$. In fact, it may be shown that the procedure for generating new candidate configuration is intrinsically biased towards those regions of $\mathcal{C}_{\text{free}}$ that have not been visited yet. Moreover, the probability that a generic configuration of $\mathcal{C}_{\text{free}}$ is added to the RRT tends to 1 as the execution time tends to infinity, provided that the configuration lies in the same connected component of $\mathcal{C}_{\text{free}}$ where the RRT is rooted.

To speed up the search for a free path going from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$, the bidirectional RRT method uses two trees T_s and T_g, respectively rooted at $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$. At each iteration, both trees are expanded with the previously described randomized mechanism. After a certain number of expansion steps, the algorithm enters a phase where it tries to connect the two trees by *extending* each one of them towards the other. This is realized by generating a $\boldsymbol{q}_{\text{new}}$ as an expansion of T_s, and trying to connect T_g to $\boldsymbol{q}_{\text{new}}$. To this end, one may modify the above expansion procedure. In particular, once $\boldsymbol{q}_{\text{new}}$ has been generated from T_s, it acts as a $\boldsymbol{q}_{\text{rand}}$ for T_g: one finds the closest configuration $\boldsymbol{q}_{\text{near}}$ in T_g, and moves from $\boldsymbol{q}_{\text{near}}$ trying to actually *reach* $\boldsymbol{q}_{\text{rand}} = \boldsymbol{q}_{\text{new}}$, hence with a variable stepsize as opposed to a constant δ.

If the segment joining $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{new}}$ is collision-free, the extension is complete and the two trees have been connected; otherwise the free portion of the segment is extracted and added to T_g together with its endpoint. At this point, T_g and T_s exchange their roles and the connection attempt is repeated. If this is not successful within a certain number of iterations, one may conclude that the two trees are still far apart and resume the expansion phase.

Like the PRM method, bidirectional RRT is probabilistically complete. A number of variations can be made on the basic scheme. For example, rather than using a constant δ for generating $\boldsymbol{q}_{\text{new}}$, one may define the stepsize as a function of the available free space, possibly going as far as $\boldsymbol{q}_{\text{rand}}$ (as in the extension procedure). This *greedy* version of the method can be much more

efficient if $\mathcal{C}$ contains extensive free regions. Moreover, RRT-based methods can be adapted to robots that do not satisfy the free-flying assumption, such as robots that are subject to nonholonomic constraints.

Extension to nonholonomic robots

Consider now the motion planning problem for a nonholonomic mobile robot whose kinematic model is expressed as (11.10). As seen in the previous chapter, admissible paths in configuration space must satisfy constraint (11.40). For example, in the case of a robot with unicycle kinematics, rectilinear paths in configuration space — such as those used in the RRT expansion to move from $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{new}}$ — are not admissible in general.

A simple yet general approach to the design of nonholonomic motion planning methods is to use *motion primitives*, i.e., a finite set of admissible local paths in configuration space, each produced by a specific choice of the velocity inputs in the kinematic model. Admissible paths are generated as a concatenation of motion primitives. In the case of a unicycle robot, for example, the following set of velocity inputs

$$v = \bar{v} \qquad \omega = \{-\bar{\omega}, 0, \bar{\omega}\} \qquad t \in [0, \Delta] \tag{12.9}$$

results in three[6] admissible local paths: the first and the third are respectively a left turn and a right turn along arcs of circle, while the second is a rectilinear path (Fig. 12.11, left).

The expansion of an RRT for a nonholonomic mobile robot equipped with a set of motion primitives is quite similar to the previously described procedure. The difference is that, once identified the configuration $\boldsymbol{q}_{\text{near}}$ on T that is closest to $\boldsymbol{q}_{\text{rand}}$, the new configuration $\boldsymbol{q}_{\text{new}}$ is generated by applying the motion primitives starting from $\boldsymbol{q}_{\text{near}}$ and choosing one of the produced configurations, either randomly or as the closest to $\boldsymbol{q}_{\text{rand}}$. Clearly, $\boldsymbol{q}_{\text{new}}$ and the admissible local path joining $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{new}}$ are subject to a collision test. Figure 12.11, right, shows an example of RRT — more precisely, its projection on the workspace — for a unicycle equipped with the motion primitives (12.9).

Under suitable assumptions, it is possible to show that if the goal configuration $\boldsymbol{q}_g$ can be reached from the start configuration $\boldsymbol{q}_s$ through a collision-free concatenation of motion primitives,[7] the probability that $\boldsymbol{q}_g$ is added to

[6] Note that these particular motion primitives include neither backward motion nor rotation on the spot. A unicycle with constant positive driving velocity v and bounded steering velocity is called *Dubins car* in the literature.

[7] This hypothesis, obviously necessary, implies that the choice of motion primitives must be sufficiently 'rich' to guarantee that the set of configuration reachable through concatenation is 'dense' enough with respect to $\mathcal{C}_{\text{free}}$. For example, the unicycle with the motion primitives (12.9) with a variable time interval Δ can reach any configuration in $\mathcal{C}_{\text{free}}$, although not necessarily with a path that is entirely contained in $\mathcal{C}_{\text{free}}$. This property is instead guaranteed if the *Reeds–Shepp curves* given by (11.56) are used as motion primitives.

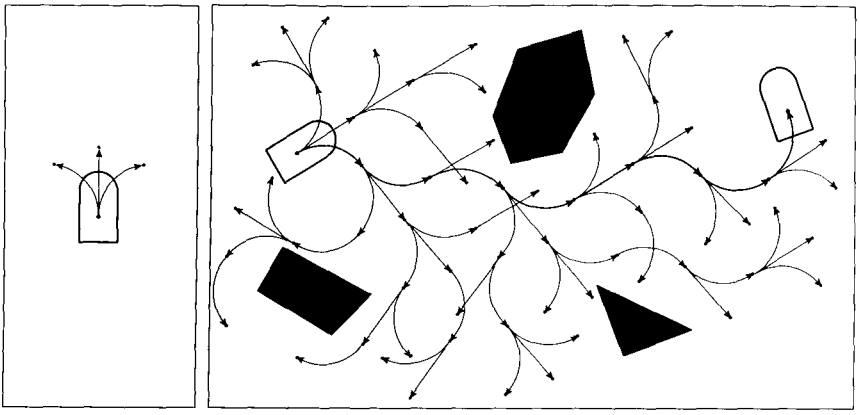

Fig. 12.11. RRT-based motion planning for a unicycle; *left*: a set of motion primitives, *right*: an example of RRT

the tree T tends to 1 as the execution time tends to infinity. To increase the efficiency of the search, also in this case it is possible to devise a bidirectional version of the method.

12.6 Planning via Artificial Potentials

All the methods so far presented are suitable for off-line motion planning, because they require a priori knowledge of the geometry and the pose of the obstacles in the robot workspace. This assumption is reasonable in many cases, e.g., when an industrial manipulator is moving in a robotized cell. However, in service robotics applications the robot must be able to plan its motion on-line, i.e., using partial information on the workspace gathered during the motion on the basis of sensor measurements.

An effective paradigm for on-line planning relies on the use of *artificial potential fields*. Essentially, the point that represents the robot in configuration space moves under the influence of a potential field U obtained as the superposition of an *attractive* potential to the goal and a *repulsive* potential from the $\mathcal{C}$-obstacle region. Planning takes place in an incremental fashion: at each robot configuration $\boldsymbol{q}$, the artificial force generated by the potential is defined as the negative gradient $-\nabla U(\boldsymbol{q})$ of the potential, which indicates the most promising direction of local motion.

12.6.1 Attractive Potential

The attractive potential is designed so as to guide the robot to the goal configuration $\boldsymbol{q}_g$. To this end, one may use a *paraboloid* with vertex in $\boldsymbol{q}_g$:

$$U_{a1}(\boldsymbol{q}) = \frac{1}{2} k_a \boldsymbol{e}^T(\boldsymbol{q}) \boldsymbol{e}(\boldsymbol{q}) = \frac{1}{2} k_a \|\boldsymbol{e}(\boldsymbol{q})\|^2, \tag{12.10}$$

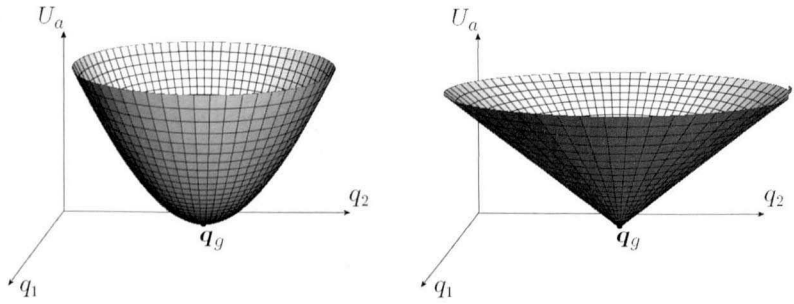

Fig. 12.12. The shape of the paraboloidal attractive potential U_{a1} (*left*) and of the conical attractive potential U_{a2} (*right*) in the case $\mathcal{C} = \mathbb{R}^2$, for $k_a = 1$

where $k_a > 0$ and $\boldsymbol{e} = \boldsymbol{q}_g - \boldsymbol{q}$ is the 'error' vector with respect to the goal configuration $\boldsymbol{q}_g$. This function is always positive and has a global minimum in $\boldsymbol{q}_g$, where it is zero. The resulting attractive force is defined as

$$\boldsymbol{f}_{a1}(\boldsymbol{q}) = -\nabla U_{a1}(\boldsymbol{q}) = k_a \boldsymbol{e}(\boldsymbol{q}). \tag{12.11}$$

Hence, $\boldsymbol{f}_{a1}$ converges linearly to zero when the robot configuration $\boldsymbol{q}$ tends to the goal configuration $\boldsymbol{q}_g$.

Alternatively, it is possible to define a *conical* attractive potential as

$$U_{a2}(\boldsymbol{q}) = k_a \|\boldsymbol{e}(\boldsymbol{q})\|. \tag{12.12}$$

Also U_{a2} is always positive, and zero in $\boldsymbol{q}_g$. The corresponding attractive force is

$$\boldsymbol{f}_{a2}(\boldsymbol{q}) = -\nabla U_{a2}(\boldsymbol{q}) = k_a \frac{\boldsymbol{e}(\boldsymbol{q})}{\|\boldsymbol{e}(\boldsymbol{q})\|}, \tag{12.13}$$

that is constant in modulus. This represents an advantage with respect to the force $\boldsymbol{f}_{a1}$ generated by the paraboloidal attractive potential, which tends to grow indefinitely as the error vector increases in norm. On the other hand, $\boldsymbol{f}_{a2}$ is indefinite in $\boldsymbol{q}_g$. Figure 12.12 shows the shape of U_{a1} and U_{a2} in the case $\mathcal{C} = \mathbb{R}^2$, with $k_a = 1$.

A choice that combines the advantages of the above two potentials is to define the attractive potential as a conical surface away from the goal and as a paraboloid in the vicinity of $\boldsymbol{q}_g$. In particular, by placing the transition between the two potentials where $\|\boldsymbol{e}(\boldsymbol{q})\| = 1$ (i.e., on the surface of the sphere of unit radius centred in $\boldsymbol{q}_g$) one obtains an attractive force that is continuous for any $\boldsymbol{q}$ (see also Problem 12.9).

12.6.2 Repulsive Potential

The repulsive potential U_r is added to the attractive potential U_a to prevent the robot from colliding with obstacles as it moves under the influence of the

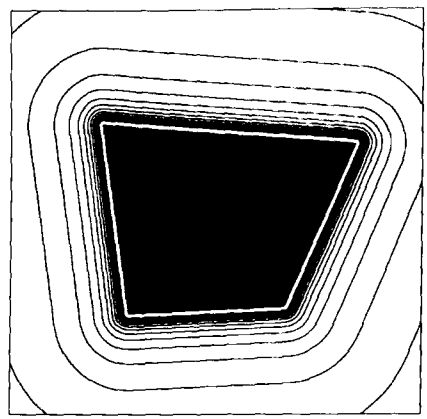

Fig. 12.13. The equipotential contours of the repulsive potential U_r in the range of influence of a polygonal $\mathcal{C}$-obstacle in $\mathcal{C} = \mathbb{R}^2$, for $k_r = 1$ and $\gamma = 2$

attractive force $\boldsymbol{f}_a$. In particular, the idea is to build a barrier potential in the vicinity of the $\mathcal{C}$-obstacle region, so as to repel the point that represents the robot in $\mathcal{C}$.

In the following, it will be assumed that the $\mathcal{C}$-obstacle region has been partitioned in convex components $\mathcal{CO}_i$, $i = 1, \ldots, p$. These components may coincide with the $\mathcal{C}$-obstacles themselves; this happens, for example, when the robot $\mathcal{B}$ is a convex polygon (polyhedron) translating with a fixed orientation in $\mathbb{R}^2$ ($\mathbb{R}^3$) among convex polygonal (polyhedral) obstacles (see Sect. 12.2.2). In the presence of non-convex $\mathcal{C}$-obstacles, however, it is necessary to perform the decomposition in convex components before building the repulsive potential.

For each convex component $\mathcal{CO}_i$, define an associated repulsive potential as

$$U_{r,i}(\boldsymbol{q}) = \begin{cases} \dfrac{k_{r,i}}{\gamma} \left(\dfrac{1}{\eta_i(\boldsymbol{q})} - \dfrac{1}{\eta_{0,i}} \right)^{\gamma} & \text{if } \eta_i(\boldsymbol{q}) \leq \eta_{0,i} \\ 0 & \text{if } \eta_i(\boldsymbol{q}) > \eta_{0,i}, \end{cases} \tag{12.14}$$

where $k_{r,i} > 0$, $\eta_i(\boldsymbol{q}) = \min_{\boldsymbol{q}' \in \mathcal{CO}_i} \|\boldsymbol{q} - \boldsymbol{q}'\|$ is the distance of $\boldsymbol{q}$ from $\mathcal{CO}_i$, $\eta_{0,i}$ is the *range of influence* of $\mathcal{CO}_i$ and $\gamma = 2, 3, \ldots$. The potential $U_{r,i}$ is zero outside and positive inside the range of influence $\eta_{0,i}$ and tends to infinity as the boundary of $\mathcal{CO}_i$ is approached, more abruptly as γ is increased (a typical choice is $\gamma = 2$).

When $\mathcal{C} = \mathbb{R}^2$ and the convex component $\mathcal{CO}_i$ is polygonal, an *equipotential contour* of $U_{r,i}$ (i.e., the locus of configurations $\boldsymbol{q}$ such that $U_{r,i}$ has a certain constant value) consists of rectilinear tracts that are parallel to the sides of the polygon, connected by arcs of circle in correspondence of the vertices, as shown in Fig. 12.13. Note how the contours get closer to each other in the proximity of the $\mathcal{C}$-obstacle boundary, due to the hyperboloidic profile

of the potential. When $\mathcal{C} = \mathbb{R}^3$ and the convex component $\mathcal{CO}_i$ is polyhedral, the *equipotential surfaces* of $U_{r,i}$ are copies of the faces of $\mathcal{CO}_i$, connected by patches of cylindrical surfaces in correspondence of the edges and spherical surfaces in correspondence of the vertices of $\mathcal{CO}_i$.

The repulsive force resulting from $U_{r,i}$ is

$$\boldsymbol{f}_{r,i}(\boldsymbol{q}) = -\nabla U_{r,i}(\boldsymbol{q}) = \begin{cases} \dfrac{k_{r,i}}{\eta_i^2(\boldsymbol{q})} \left(\dfrac{1}{\eta_i(\boldsymbol{q})} - \dfrac{1}{\eta_{0,i}} \right)^{\gamma-1} \nabla \eta_i(\boldsymbol{q}) & \text{if } \eta_i(\boldsymbol{q}) \leq \eta_{0,i} \\ 0 & \text{if } \eta_i(\boldsymbol{q}) > \eta_{0,i}. \end{cases} \tag{12.15}$$

Denote by $\boldsymbol{q}_m$ the configuration of $\mathcal{CO}_i$ that is closer to $\boldsymbol{q}$ ($\boldsymbol{q}_m$ is uniquely determined in view of the convexity of $\mathcal{CO}_i$). The gradient vector $\nabla \eta_i(\boldsymbol{q})$, which is orthogonal to the equipotential contour (or surface) passing through $\boldsymbol{q}$, is directed as the half-line originating from $\boldsymbol{q}_m$ and passing through $\boldsymbol{q}$. If the boundary of $\mathcal{CO}_i$ is piecewise differentiable, function η_i is differentiable everywhere in $\mathcal{C}_{\text{free}}$ and $\boldsymbol{f}_{r,i}$ is continuous in the same space.[8]

The aggregate repulsive potential is obtained by adding up the individual potentials associated with the convex components of $\mathcal{CO}$:

$$U_r(\boldsymbol{q}) = \sum_{i=1}^{p} U_{r,i}(\boldsymbol{q}). \tag{12.16}$$

If $\eta_i(\boldsymbol{q}_g) \geq \eta_{0,i}$ for $i = 1, \ldots, p$ (i.e., if the goal is placed outside the range of influence of each obstacle component $\mathcal{CO}_i$), the value of the aggregate repulsive field U_r is zero in $\boldsymbol{q}_g$. In the following, it will be assumed that this is the case.

12.6.3 Total Potential

The total potential U_t is obtained by superposition of the attractive and the aggregate repulsive potentials:

$$U_t(\boldsymbol{q}) = U_a(\boldsymbol{q}) + U_r(\boldsymbol{q}). \tag{12.17}$$

This results in the force field

$$\boldsymbol{f}_t(\boldsymbol{q}) = -\nabla U_t(\boldsymbol{q}) = \boldsymbol{f}_a(\boldsymbol{q}) + \sum_{i=1}^{p} \boldsymbol{f}_{r,i}(\boldsymbol{q}). \tag{12.18}$$

[8] Note the relevance in this sense of the assumption that the component $\mathcal{CO}_i$ is convex. If it were otherwise, there would exist configurations in $\mathcal{C}_{\text{free}}$ for which $\boldsymbol{q}_m$ would not be uniquely defined. In these configurations, belonging by definition to the generalized Voronoi diagram $\mathcal{V}(\mathcal{C}_{\text{free}})$, function η_i would not be differentiable, resulting in a discontinuous repulsive force. This might induce undesired effects on the planned path (for example, oscillations).

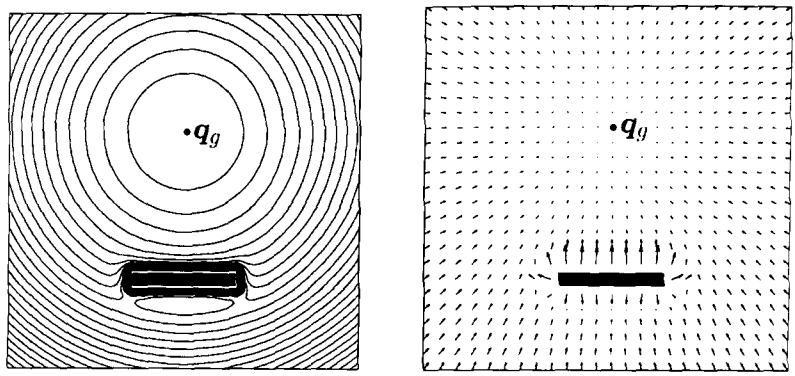

Fig. 12.14. The total potential in $\mathcal{C} = \mathbb{R}^2$ obtained by superposition of a hyperboloidic attractive potential and a repulsive potential for a rectangular $\mathcal{C}$-obstacle: *left*: the equipotential contours, *right*: the resulting force field

U_t clearly has a global minimum in $\boldsymbol{q}_g$, but there may also exist some *local minima* where the force field is zero. Considering for simplicity the case $\mathcal{C} = \mathbb{R}^2$, this happens in the 'shadow zone' of a $\mathcal{C}$-obstacle when the repulsive potential $U_{r,i}$ has equipotential contours with lower curvature (e.g., segments) than the attractive potential in the same area. See for example Fig. 12.14. where a local minimum is clearly present 'below' the $\mathcal{C}$-obstacle. A remarkable exception is the case (*sphere world*) in which all the convex components $\mathcal{CO}_i$ of the $\mathcal{C}$-obstacle region are spheres. In this situation, the total potential exhibits isolated saddle points (where the force field is still zero) but no local minima.

12.6.4 Planning Techniques

There are three different approaches for planning collision-free motions on the basis of a total artificial potential U_t and the associated force field $\boldsymbol{f}_t = -\nabla U_t$. They are briefly discussed below:

1. The first possibility is to let

$$\boldsymbol{\tau} = \boldsymbol{f}_t(\boldsymbol{q}), \tag{12.19}$$

   hence considering $\boldsymbol{f}_t(\boldsymbol{q})$ as a vector of generalized forces that induce a motion of the robot in accordance with its dynamic model.
2. The second method regards the robot as a unit point mass moving under the influence of $\boldsymbol{f}_t(\boldsymbol{q})$, as in

$$\ddot{\boldsymbol{q}} = \boldsymbol{f}_t(\boldsymbol{q}). \tag{12.20}$$

3. The third possibility is to interpret the force field $\boldsymbol{f}_t(\boldsymbol{q})$ as a desired velocity for the robot, by letting

$$\dot{\boldsymbol{q}} = \boldsymbol{f}_t(\boldsymbol{q}). \tag{12.21}$$

In principle, one could use these three approaches for on-line as well as off-line motion planning. In the first case, (12.19) directly represents control inputs for the robot, whereas the implementation of (12.20) requires the solution of the inverse dynamics problem, i.e., the substitution of $\ddot{\boldsymbol{q}}$ in the robot dynamic model to compute the generalized forces $\boldsymbol{\tau}$ that realize such accelerations. Equation (12.21) can instead be used on-line in a kinematic control scheme, in particular to provide the reference inputs for the low-level controllers that are in charge of reproducing such generalized velocities as accurately as possible. In any case, the artificial force field $\boldsymbol{f}_t$ represents, either directly or indirectly, a true feedback control that guides the robot towards the goal, while trying at the same time to avoid collisions with the workspace obstacles that have been detected by the sensory system. To emphasize this aspect, on-line motion generation based on artificial potentials is also referred to as *reactive* planning.

In off-line motion planning, configuration space paths are generated by simulation, i.e., integrating numerically the robot dynamic model if (12.19) is used, or directly by the differential equations (12.20) and (12.21).

In general, the use of (12.19) generates smoother paths, because with this scheme the reactions to the presence of obstacles are naturally 'filtered' through the robot dynamics. On the other hand, the strategy represented by (12.21) is faster in executing the motion corrections suggested by the force field $\boldsymbol{f}_t$, and may thus be considered safer. The characteristics of scheme (12.20) are clearly intermediate between the other two. Another aspect to be considered is that using (12.21) guarantees (in the absence of local minima) the asymptotic stability of $\boldsymbol{q}_g$ (i.e., the robot reaches the goal with zero velocity), whereas this is not true for the other two motion generation strategies. To achieve asymptotic stability with (12.19) and (12.20), a damping term proportional to the robot velocity $\dot{\boldsymbol{q}}$ must be added to $\boldsymbol{f}_t$.

In view of the above discussion, it is not surprising that the most common choice is the simple numerical integration of (12.21) via the Euler method:

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + T\boldsymbol{f}_t(\boldsymbol{q}_k), \tag{12.22}$$

where $\boldsymbol{q}_k$ and $\boldsymbol{q}_{k+1}$ represent respectively the current and the next robot configuration, and T is the integration step. To improve the quality of the generated path, it is also possible to use a variable T, smaller when the modulus of the force field $\boldsymbol{f}_t$ is larger (in the vicinity of obstacles) or smaller (close to the destination $\boldsymbol{q}_g$). Recalling that $\boldsymbol{f}_t(\boldsymbol{q}) = -\nabla U_t(\boldsymbol{q})$, Eq. (12.22) may be easily interpreted as a numerical implementation of the *gradient method* for the minimization of $U_t(\boldsymbol{q})$, often referred to as the *algorithm of steepest descent*.

12.6.5 The Local Minima Problem

Whatever technique is used to plan motions on the basis of artificial potentials, local minima of U_t — where the total force field $\boldsymbol{f}_t$ is zero — represent a

problem. For example, if (12.22) is used and the generated path enters the basin of attraction of a local minimum, the planning process is bound to stop there, without reaching the goal configuration.[9] Actually, the same problem may occur also when (12.19) or (12.20) are used if the basin of attraction of the local minimum is sufficiently large. On the other hand, as noticed previously, the total potential U_t obtained by the superposition of an attractive and a repulsive potential invariably exhibits local minima, except for very particular cases. This means that motion planning methods based on artificial potentials are not complete in general, because it may happen that the goal configuration $\boldsymbol{q}_g$ is not reached even though a solution exists.

Best-first algorithm

A simple workaround for the local minima problem is to use a *best-first* algorithm, which is formulated under the assumption that the free configuration space $\mathcal{C}_{\text{free}}$ has been discretized using a regular grid. In general, the discretization procedure results in the loss of some boundary regions of $\mathcal{C}_{\text{free}}$, which are not represented as free in the gridmap. Each free cell of the grid is assigned the value of the total potential U_t computed at the centroid of the cell. Planning proceeds by building a tree T rooted at the start configuration $\boldsymbol{q}_s$. At each iteration, the leaf with the minimum value of U_t is selected and its adjacent[10] cells are examined. Those that are not already in T are added as children of the considered leaf. Planning is successful when the cell containing the goal is reached (in this case, the solution path is built by tracing back the arcs of the tree from $\boldsymbol{q}_g$ to $\boldsymbol{q}_s$), whereas failure is reported when all the cells accessible from $\boldsymbol{q}_s$ have been explored without reaching $\boldsymbol{q}_g$.

The above best-first algorithm evolves as a grid-discretized version of the algorithm of steepest descent (12.22), until a cell is reached which represents a minimum for the associated total potential. If it is a local minimum, the algorithm visits ('fills') its entire basin of attraction and finally leaves it, reaching a point from which planning can continue. The resulting motion planning method is resolution complete, because a solution is found only if one exists on the gridmap that represents $\mathcal{C}_{\text{free}}$ by defect. In general, increasing the grid resolution may be necessary to recover the possibility of determining a solution. In any case, the time complexity of the basin filling procedure, which is exponential in the dimension of $\mathcal{C}$ because such is the number of adjacent cells to a given one, makes the best-first algorithm applicable only in configuration spaces of low dimension (typically, not larger than 3).

---

[9] The probability of reaching a saddle point instead is extremely low; besides, any perturbation would allow the planner to exit such a point.

[10] Various definitions of adjacency may be adopted. For example, in $\mathbb{R}^2$ one may use 1-adjacency or 2-adjacency. In the first case, each cell c has four adjacent cells, while in the second they are eight. The resulting paths will obviously reflect this fact.

A more effective approach is to include in the best-first method a randomized mechanism for evading local minima. In practice, one implements a version of the best-first algorithm in which the number of iterations aimed at filling the basin of attraction of local minima is bounded. When the bound is reached, a sequence of random steps (*random walk*) is taken. This usually allows the planner to evade the basin in a much shorter time than would be needed by the complete filling procedure. As the probability of evading the basin of attraction of a local minimum approaches 1 when the number of random steps tends to infinity, this *randomized best-first* method is probabilistically complete (in addition to being resolution complete).

Navigation functions

Although the best-first algorithm (in its basic or randomized version) represents a solution to the problem of local minima, it may generate paths that are very inefficient. In fact, with this strategy the robot will still enter (and then evade) the basin of attraction of any local minimum located on its way to goal. A more radical approach is based on the use of *navigation functions*, i.e., artificial potentials that have no local minima. As already mentioned, if the $\mathcal{C}$-obstacles are spheres this property already holds for the potential U_t defined by superposition of an attractive and a repulsive field. A first possibility would then be to approximate by excess all $\mathcal{C}$-obstacles with spheres, and use the total potential U_t. Clearly, such an approximation may severely reduce the free configuration space $\mathcal{C}_{\text{free}}$, and even destroy its connectedness; for example, imagine what would happen in the case depicted in Fig. 12.4.

In principle, a mathematically elegant way to define a navigation function consists of building first a differentiable homeomorphism (a *diffeomorphism*) that maps the $\mathcal{C}$-obstacle region to a collection of spheres, then generating a classical total potential in the transformed space, and finally mapping it back to the original configuration space so as to obtain a potential free of local minima. If the $\mathcal{C}$-obstacles are *star-shaped*,[11] such a diffeomorphism actually exists, and the procedure outlined above provides in fact a navigation function. Another approach is to build the potential using *harmonic functions*, that are the solutions of a particular differential equation that describes the physical process of heat transmission or fluid dynamics.

Generating a navigation function is, however, computationally cumbersome, and the associated planning methods are thus mainly of theoretical interest. A notable exception, at least in low-dimensional configuration spaces, is the *numerical navigation function*. This is a potential built on a gridmap representation of $\mathcal{C}_{\text{free}}$ by assigning value 0 to the cell containing $\boldsymbol{q}_g$, value 1 to its adjacent cells, value 2 to the unvisited cells among those adjacent to

[11] A subset $S \subset \mathbb{R}^n$ is said to be star-shaped if it is homeomorphic to the closed unit sphere in $\mathbb{R}^n$ and has a point p (*centre*) such that any other point in S may be joined to p by a segment that is entirely contained in S.

2	1	2	3	4	5	6	7	8	9		19
1	0	1			6	7	8	9	10		18
2	1	2	3		7	8		10	11		17
3		3	4	5	6	7	8		12		16
4			5	6	7			12	13		15
5	6	7	6	7	8	9	10	11	12	13	14
6	7	8	7	8	9	10	11	12	13	14	15

Fig. 12.15. An example of numerical navigation function in a simple two-dimensional gridmap using 1-adjacency. Cells in gray denote a particular solution path obtained by following from the start (cell 12) the steepest descent of the potential on the gridmap

cells with potential 1, and so on. To understand better the procedure, one may visualize a wavefront that originates at $\boldsymbol{q}_g$ and expands according to the adopted adjacency definition (*wavefront expansion algorithm*). It is easy to realize that the obtained potential is free of local minima, and therefore its use in conjunction with the algorithm of steepest descent provides a motion planning method that is complete in resolution (see Fig. 12.15).

Finally, it should be mentioned that the use of navigation functions is limited to off-line motion planning, because their construction — be they continuous or discrete in nature — requires the a priori knowledge of the geometry and pose of the workspace obstacles. As for on-line motion planning, in which the obstacles are gradually reconstructed via sensor measurements as the robot moves, the incremental construction of a total potential by superposition of attractive and repulsive fields represents a simple, often effective method to generate collision-free motions, even though completeness cannot be claimed. In any case, motion planning based on artificial potentials belong to the single-query category, because the attractive component of the potential depends on the goal configuration $\boldsymbol{q}_g$.

12.7 The Robot Manipulator Case

Generating collision-free movements for robot manipulators is a particularly important category of motion planning problems. In general, the computational complexity associated with this problem is substantial, due to the high dimension of the configuration space (typically $n \geq 4$) and to the presence of rotational DOFs (revolute joints).

It is sometimes possible to reduce the dimension of the configuration space $\mathcal{C}$ approximating by excess the size of the robot. For example, in a six-DOF anthropomorphic manipulator one can replace the last three links of the kinematic chain (the spherical wrist) and the end-effector with the volume they

'sweep' when the corresponding joints move across their whole available range. The dimension of the configuration space becomes three, as planning concerns only the base, shoulder and elbow joints, while the wrist can move arbitrarily. Clearly, this approximation is conservative, and hence acceptable only if the aforementioned volume is small with respect to the workspace of the manipulator.

In the presence of rotational DOFs, the other complication is the shape of the $\mathcal{C}$-obstacles, which is complex even for simple workspace obstacles due to the strong nonlinearity introduced by the manipulator inverse kinematics (recall Fig. 12.4). Apart from the intrinsic difficulty of computing $\mathcal{C}$-obstacles, their non-polyhedral shape does not allow the application of the planning methods presented in Sects. 12.3 and 12.4.

The most convenient choice for off-line planning is represented by probabilistic methods, which exhibit the best performance in high-dimensional configuration spaces. Moreover, as the computation of the $\mathcal{C}$-obstacles is not required, these planners are not affected by their shape. However, it should be noted that collision checking — which is an essential tool for probabilistic motion planning — becomes more onerous as the number of DOFs is increased.

For on-line planning, the best results are obtained by a suitable adaptation of the method based on artificial potentials. In particular, to avoid the computation of $\mathcal{C}$-obstacles and at the same time plan in a space of reduced dimension, the potential is directly built in the workspace $\mathcal{W} = \mathbb{R}^N$ rather than in the configuration space $\mathcal{C}$, and acts on a set of *control points* located on the manipulator. Among these is included a point that represents the end-effector (to which is assigned the goal of the motion planning problem) and at least one point (possibly variable in time) for each body of the linkage. While the attractive potential only influences the end-effector representative point, the repulsive potential acts on all control points. As a consequence, the artificial potentials used in this scheme are actually two: an attractive-repulsive field for the end-effector, and a repulsive field for the other control points distributed on the manipulator links.

As before, different approaches may be used to convert the force fields generated by the artificial potentials to commands for the manipulator. Denote by $\boldsymbol{p}_i(\boldsymbol{q})$, $i = 1, \ldots, P$, the coordinates in $\mathcal{W}$ of the P control points in correspondence of the configuration $\boldsymbol{q}$ of the manipulator. In particular, $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{P-1}$ are the control points located on the manipulator links, subject only to the repulsive potential U_r, while $\boldsymbol{p}_P$ is the control point for the end-effector, which is subject to the total potential $U_t = U_a + U_r$.

A first possibility is to impose to the robot joints the generalized forces which would result from the combined action of the various force fields acting on the control points in the workspace, according to

$$\boldsymbol{\tau} = -\sum_{i=1}^{P-1} \boldsymbol{J}_i^T(\boldsymbol{q}) \nabla U_r(\boldsymbol{p}_i) - \boldsymbol{J}_P^T(\boldsymbol{q}) \nabla U_t(\boldsymbol{p}_P), \tag{12.23}$$

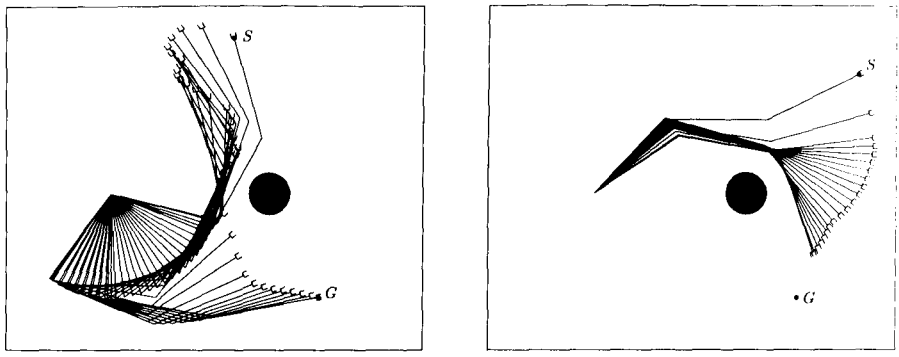

Fig. 12.16. Examples of motion planning via artificial potentials acting on control points for a planar 3R manipulator; *left*: planning is successful and leads to a collision-free motion between the start S and the goal G, *right*: a failure is reported because the manipulator is stuck at a force equilibrium

where $\boldsymbol{J}_i(\boldsymbol{q})$, $i = 1, \ldots, P$, denotes the Jacobian of the direct kinematics function associated with the control point $\boldsymbol{p}_i(\boldsymbol{q})$.

Alternatively, a purely kinematic planning scheme is obtained by letting

$$\dot{\boldsymbol{q}} = -\sum_{i=1}^{P-1} \boldsymbol{J}_i^T(\boldsymbol{q}) \nabla U_r(\boldsymbol{p}_i) - \boldsymbol{J}_P^T(\boldsymbol{q}) \nabla U_t(\boldsymbol{p}_P) \tag{12.24}$$

and feeding these joint velocities to the low-level control loops as reference signals. Note that Eq. (12.24) represents a gradient-based minimization step in the configuration space $\mathcal{C}$ of a combined potential defined in the workspace $\mathcal{W}$. In fact, a potential function acting on a control point in the workspace may be seen as a composite function of $\boldsymbol{q}$ through the associated direct kinematics relationship, and the Jacobian transpose $\boldsymbol{J}_i^T(\boldsymbol{q})$, $i = 1, \ldots, P$, maps a gradient in $\mathcal{W}$ to a gradient in $\mathcal{C}$. In formulae:

$$\nabla_{\boldsymbol{q}} U(\boldsymbol{p}_i) = \left(\frac{\partial U(\boldsymbol{p}_i(\boldsymbol{q}))}{\partial \boldsymbol{q}} \right)^T = \left(\frac{\partial U(\boldsymbol{p}_i)}{\partial \boldsymbol{p}_i} \frac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{q}} \right)^T = \boldsymbol{J}_i^T(\boldsymbol{q}) \nabla U(\boldsymbol{p}_i),$$

for $i = 1, \ldots, P$.

The above two schemes can be considered respectively the transposition of (12.19) and (12.21), and therefore they inherit the same characteristics. In particular, when (12.23) is used the motion corrections prescribed by the force fields are filtered through the dynamic model of the manipulator, and smoother movements can be expected. The kinematic scheme (12.24) instead is faster in realizing such corrections.

Finally, it should be mentioned that the use of artificial potentials that are defined in the workspace may aggravate the local minima problem. In fact, the various forces (either purely attractive or repulsive-attractive) acting on the control points may neutralize each other at the joint level, blocking the

manipulator in a configuration (*force equilibrium*) where no control point is at a local minimum of the associated potential (see Fig. 12.16). As consequence, it is always advisable to use workspace potential fields in conjunction with a randomized best-first algorithm.

Bibliography

In the last three decades, the literature on motion planning has grown considerably, and this area may now be considered as a scientific discipline in itself. In the following, only a short list is given of the seminal works for the material presented in this chapter.

The systematic use of the concept of configuration space for motion planning was proposed in [138]. The method based on the retraction of the free configuration space on the generalized Voronoi diagram was originally introduced for a robot of circular shape in [170]. The technique based on trapezoidal cell decomposition is described in [122], while [197] and [33] are among the general planning methods via decomposition mentioned at the end of Sect. 12.4.1. The approach based on approximate cell decomposition was proposed in [138]. The PRM method for probabilistic planning was introduced in [107], while the RRT method with its variants is described in [125].

The use of artificial potentials for on-line motion planning was pioneered in [113]. The concept of navigation function was introduced in [185], while its numerical version on a gridmap was described in [17], together with the best-first algorithm, also in its randomized version.

For other aspects of the motion planning problem that are merely hinted at (nonholonomic motion planning) or simply ignored (managing uncertainty, mobile obstacles) in this chapter, the reader can consult many excellent books, going from the classical treatment in [122], through [123], to the most recent texts [45, 145, 124].

Problems

12.1. Describe the nature (including the dimension) of the configuration space for a mobile manipulator consisting of a unicycle-like vehicle carrying a six-DOF anthropomorphic arm, providing a choice of generalized coordinates for the system.

12.2. With reference to a 2R manipulator, modify the definition (12.2) of configuration space distance so as to take into account the fact that the manipulator posture does not change if the joint variables q_1 and q_2 are increased (or decreased) by a multiple of 2π.

12.3. Consider a polygonal robot translating at a fixed orientation in $\mathbb{R}^2$ among polygonal obstacles. Build an example showing that the same $\mathcal{C}$-obstacle region may correspond to robot and obstacles of different shapes.

12.4. With reference to the second workspace shown in Fig. 12.4, give the numerical value of three configurations of the manipulator that lie in the three connected components of $\mathcal{C}_{\text{free}}$. Moreover, sketch the manipulator posture for each of these configurations.

12.5. Discuss the basic steps of an algorithm for computing the generalized Voronoi diagram of a limited polygonal subset of $\mathbb{R}^2$. [*Hint*: a simple algorithm is obtained by considering all the possible side-side, side-vertex and vertex-vertex pairs, and generating the elementary arcs of the diagram on the basis of the intersections of the associated equidistance contours.]

12.6. For the motion planning method via exact cell decomposition, give an example in which the path obtained as a broken line joining the midpoints of the common boundary of the channel cells goes through a vertex of the $\mathcal{C}$-obstacle region, and propose an alternative procedure for extracting a free path from the channel. [*Hint*: build a situation in which the channel from c_s to c_g contains a cell for which the entrance and the exit boundary lie on the same side.]

12.7. Implement in a computer program the PRM method for a 2R planar robot moving among circular workspace obstacles. The program receives as input a geometrical description of the obstacles (centre and radius of each obstacle) as well as the start and goal configurations, and terminates after a maximum number of iterations. If a solution path has been found, it is given as output together with the associated PRM; otherwise, a failure is reported. Discuss the performance of the method with respect to the choice of configuration space distance (for example, (12.1) or (12.2)).

12.8. Implement in a computer program the RRT method for a circular-shaped unicycle moving among square obstacles, with the motion primitives defined as in (12.9). The program receives as input the start and goal configurations, in addition to a geometrical description of the obstacles (centre and side of each obstacle), and terminates after a maximum number of iterations. If a solution path has been found, it is given as output together with the associated RRT; otherwise, a failure is reported. Build a situation in which the method cannot find a solution because of the limitations inherent to the chosen motion primitives.

12.9. For the case $\mathcal{C} = \mathbb{R}^2$, build a continuously differentiable attractive potential having a paraboloidal profile inside the circle of radius ρ and a conical profile outside. [*Hint*: modify the expression (12.12) of the conical potential using a different constant k_b in place of k_a, which already characterizes the paraboloidal potential.]

12.10. For the case $\mathcal{C} = \mathbb{R}^2$, prove that the total potential U_t may exhibit a local minimum in areas where the equipotential contours of the repulsive potential U_r have lower curvature than those of the attractive potential U_a. [*Hint*: consider a polygonal $\mathcal{C}$-obstacle and use a geometric construction.]

12.11. Consider a point robot moving in a planar workspace containing three circular obstacles, respectively of radius 1, 2 and 1 and centre in $(2,1)$, $(-1,3)$ and $(1,-2)$. The goal is the origin of the workspace reference frame. Build the total potential U_t resulting from the superposition of the attractive potential U_a to the goal and the repulsive potential U_r from the obstacles, and derive the corresponding artificial force field. Moreover, compute the coordinates of the saddle points of U_t.

12.12. Discuss the main issues arising from the application of the artificial potential technique for planning on-line the motion of an omnidirectional circular robot. Assume that the robot is equipped with a rotating laser range finder placed at its centre that measures the distance between the sensor and the closest obstacles along each direction. If the distance is larger than the maximum measurable range R, the sensor returns R as a reading. Sketch a possible extension of the method to a unicycle-like mobile robot.

12.13. Implement in a computer program the motion planning method based on the numerical navigation function. The program receives as input a two-dimensional gridmap, in which some of the cells are labelled as 'obstacles', with the specification of the start and the goal cells. If the algorithm is successful, a sequence of free cells from the start to the goal is provided as output. With the aid of some examples, compare the average length of the solution paths obtained using 1-adjacency and 2-adjacency to build the navigation function.