Chapter 1

Introduction and Overview

A robot is a versatile mechanical device — for example, a manipulator arm, a multi-joint multi-fingered hand, a wheeled or legged vehicle, a free-flying platform, or a combination of these — equipped with actuators and sensors under the control of a computing system. It operates in a workspace within the real world. This workspace is populated by physical objects and is subject to the laws of nature. The robot performs tasks by executing motions in the workspace.

In this book we are interested in giving the robot the capability of planning its own motions, i.e. deciding automatically what motions to execute in order to achieve a task specified by initial and goal spatial arrangements of physical objects. Creating autonomous robots is a major undertaking in Robotics. It definitively requires that the ability to plan motions automatically be developed. Indeed, except in limited and carefully engineered environments, it is not realistic to anticipate and explicitly describe to the robot all the possible motions that it may have to execute in order to accomplish requested tasks. Even in those cases where such a description is feasible, it would certainly be useful to incorporate automatic motion planning tools in off-line robot programming

J.-C. Latombe, *Robot Motion Planning*

systems. This would allow the user to specify tasks more declaratively, by stating *what* he/she wants done rather than *how* to do it. As robots become more dexterous, the need for motion planning tools will become more critical.

A still widespread view is that motion planning is essentially some sort of simple collision checking. Motion planning is much more than that. As we will see in this chapter, it presents an unexpected variety of facets, the simplest of which raise provably difficult computational issues. In fact, the kind of operative intelligence that people use unconsciously to interact with their environment, which is needed for perception and motion planning, turns out to be extremely difficult to duplicate in a computer program.

This chapter is both an introduction to robot motion planning and an overview of the rest of the book.

We first illustrate the various aspects of motion planning (Section 1). This illustration motivates our next step, the definition of a "basic problem" which is in some way the simplest useful motion planning problem that we may investigate (Section 2). Using this problem, we present the concept of configuration space, which is a unifying concept throughout the book (Section 3) and we outline the principles of the main computational approaches to motion planning (Section 4). We then introduce the extensions of the basic problem which will be studied in greater detail later (Section 5).

An important issue in motion planning is the computational complexity of the algorithms. In the following chapters of this book, we will analyze the complexity of particular planning methods. However, it is useful to have some preliminary idea of the intrinsic complexity of the problems which we are addressing. During the last few years, significant results have been obtained on this topic. We review them in Section 6. We also analyze the impact of these results on the design of operational systems, by proposing various guidelines for simplifying problems and implementing planners of increased practical efficiency (Section 7).

Motion planning is one major problem in the creation of autonomous robots. It is not the only one. It interacts with other important problems, including real-time motion control, sensing and task planning. We briefly discuss these interactions in Section 8.

The final section of this chapter reviews some key publications (Section 9). It is an historical account of the development of the motion planning field from the late 60's to the present time. Of course, other publications not mentioned in that section have also brought significant contributions to the field. We will reference many of them later in the book.

1 Aspects of Motion Planning

Consider a robot arm whose task is to build an assembly from a set of individual mechanical parts. Assume that the task is described as both a set of models of the parts, including their geometry and perhaps some physical properties, and a sequence of spatial relations [Latombe, 1984]. Figure 1 illustrates this description with a simple construction task. The sequence determines the order in which the parts are to be assembled. Each relation describes the position of a new part with respect to the current subassembly. The automatic transformation of this sequence into a description of the robot motions to be executed is one possible application of the methods presented in the book.

Let us analyze the example in more detail. In order to assemble the parts according to the input relations, the robot has to successively grasp each part, transfer it to the current subassembly, and position it on the subassembly:

- **Grasping** the part requires the position of the robot's gripper on the object to be selected and a path to this position to be generated. The grasp position must be accessible; it must also be stable and robust enough to resist some external forces. Sometimes, a satisfactory grasping position requires the part to be grasped, put down, and re-grasped.

- **Transferring** the part requires the geometry of a path for the arm to be computed. Since the motion will typically be executed at high speed, the path should avoid any contact with obstacles. Due to imprecision in motion control, it may even have to guarantee some minimal clearance.

- **Positioning** the part on the subassembly (part mating) requires motion commands to be selected for achieving a goal with high precision in a very constrained space. In general, due to uncertainty, a single

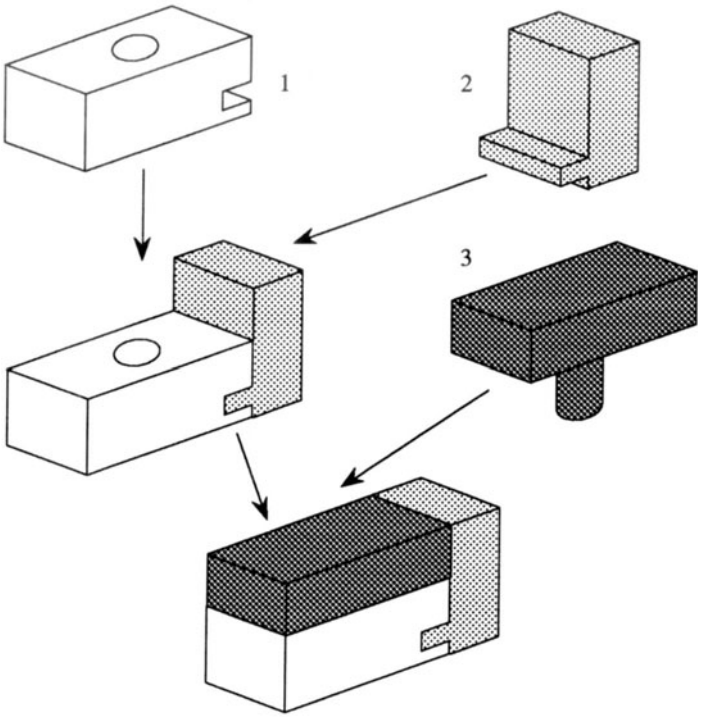

Figure 1. This simple construction task is defined by both the models of three constituent parts and a sequence of two spatial relations. Each relation may be input as a set of primitive relations among geometrical features, e.g. "Face A of Object 1 is against Face B of Object 2", "Shaft C of Object 3 fits Hole D of Object 1" [Ambler and Popplestone, 1975] [Mazer, 1983]. It may then be converted into a quantitative internal representation, e.g. the "homogeneous coordinate matrix" describing the transform between two Cartesian coordinate frames attached to the objects [Craig, 1986].

motion command is not enough; instead, a motion strategy involving several commands and using sensory inputs is necessary.

Prior knowledge of the spatial arrangement of the robot workspace is needed to plan these operations. For example, planning a collision-free path requires knowing where the obstacles are located. If complete knowledge is available (it may have been provided by the user or by a computer-aided design system, or obtained through sensing), motion

plans can be fully generated and then executed. If this is not the case, some interweaving of planning and execution is necessary. When uncertainty is well-bounded (this is usually the case during part mating), robust plans anticipating various contingencies may be suitable.

The above example can easily be made more complicated. For instance, there might be more than one robot to achieve the task, or there might be mobile obstacles in the workspace. Then, some motion coordination is necessary, implying that motions are planned not just as geometrical paths, but as functions of time. A robot might also be equipped with a dexterous multi-joint multi-fingered hand allowing complex types of motions within the hand. The task might include some performance criterion to optimize, e.g. minimize the total amount of time needed to perform the task. Rather than specifying a sequence of spatial relations, the input description might only describe the spatial arrangement of the parts in the goal assembly, leaving to the planner the generation of the appropriate sequence.

The example stresses the fact that motion planning does not consist of a single, well-delimited problem. Instead, it looks more like a collection of many problems, which are more or less variants of each other. Although one may envision that a general formulation covering all possible cases could be ultimately established, such a formulation is certainly beyond our current understanding of the conceptual and computational issues underlying motion planning.

In order to organize the book, we will first consider a "basic motion planning problem" which deals with geometrical issues only, and then several extensions of this problem which involve more sophisticated geometrical and non-geometrical issues. The basic problem is defined in Section 2 and several extensions are presented in Section 5.

2 Basic Problem

The goal of defining a basic motion planning problem is to isolate some central issues and investigate them in depth before considering additional difficulties.

In the basic problem, we assume that the robot is the only moving object in the workspace and we ignore the dynamic properties of the robot,

thus avoiding temporal issues. We also restrict motions to non-contact motions, so that the issues related to the mechanical interaction between two physical objects in contact can be ignored. These assumptions essentially transform the "physical" motion planning problem into a purely geometrical path planning problem. We simplify even further the geometrical issues by assuming that the robot is a single rigid object, i.e. an object whose points are fixed with respect to each other. The motions of this object are only constrained by the obstacles.

The **basic motion planning problem** resulting from these simplifications is the following:

> Let $\mathcal{A}$ be a single rigid object — the *robot* — moving in a Euclidean space $\mathcal{W}$, called *workspace*, represented as $\mathbf{R}^N$, with $N = 2$ or 3.
>
> Let $\mathcal{B}_1, ..., \mathcal{B}_q$ be fixed rigid objects distributed in $\mathcal{W}$. The $\mathcal{B}_i$'s are called *obstacles*.
>
> Assume that both the geometry of $\mathcal{A}, \mathcal{B}_1, ..., \mathcal{B}_q$ and the locations of the $\mathcal{B}_i$'s in $\mathcal{W}$ are accurately known. Assume further that no kinematic constraints limit the motions of $\mathcal{A}$ (we say that $\mathcal{A}$ is a *free-flying object*).
>
> The problem is: Given an initial position and orientation and a goal position and orientation of $\mathcal{A}$ in $\mathcal{W}$, generate a *path* τ specifying a continuous sequence of positions and orientations of $\mathcal{A}$ avoiding contact with the $\mathcal{B}_i$'s, starting at the initial position and orientation, and terminating at the goal position and orientation. Report failure if no such path exists.

Figure 2 illustrates two instances of the basic motion planning problem. It shows the paths of a rectangle (Figure 2.a) and an L-shaped polygon (Figure 2.b) among polygonal obstacles in a two-dimensional workspace.

Of course, the basic problem is somewhat oversimplified. We will see that it is nonetheless a difficult problem and that several of its solutions have straightforward extensions to more involved problems. Furthermore, solving it is also of practical interest. For instance, some mobile robot navigation problems can be realistically expressed as instances of the basic problem.

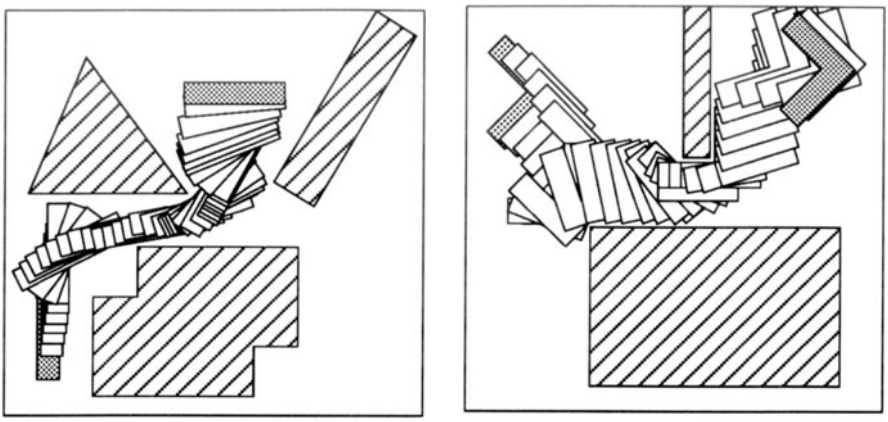

Figure 2. This figure illustrates the basic motion planning problem with two robots — a rectangle in Figure a and an L-shaped polygon in Figure b. The contact-free paths are displayed as discrete sequences of positions and orientations of the robot.

Although intuitively simple, some aspects of the above definition are still too informal and must be made more precise. For instance, the concept of a "free-flying" object is probably not very clear at this point. One possible formal statement of the problem is known as the *configuration space formulation.*

3 Configuration Space Formulation

The underlying idea of configuration space is to represent the robot as a point in an appropriate space — the robot's configuration space — and to map the obstacles in this space. This mapping transforms the problem of planning the motion of a dimensioned object into the problem of planning the motion of a point. It makes the constraints on the motions of the robot more explicit.

3.1 Notion of Configuration Space

Consider the basic problem. Let the robot $\mathcal{A}$ (at a certain position and orientation) be described as a compact (i.e. closed and bounded) subset

of $\mathcal{W} = \mathbf{R}^N$, $N = 2$ or 3, and the obstacles $\mathcal{B}_1, ..., \mathcal{B}_q$ be closed subsets of $\mathcal{W}$. In addition, let $\mathcal{F}_\mathcal{A}$ and $\mathcal{F}_\mathcal{W}$ be Cartesian frames embedded in $\mathcal{A}$ and $\mathcal{W}$, respectively. $\mathcal{F}_\mathcal{A}$ is a moving frame, while $\mathcal{F}_\mathcal{W}$ is a fixed one. By definition, since $\mathcal{A}$ is rigid, every point a of $\mathcal{A}$ has a fixed position with respect to $\mathcal{F}_\mathcal{A}$. But a's position in $\mathcal{W}$ depends on the position and orientation of $\mathcal{F}_\mathcal{A}$ relative to $\mathcal{F}_\mathcal{W}$. Since the $\mathcal{B}_i$'s are both rigid and fixed in $\mathcal{W}$, every point of $\mathcal{B}_i$, for all $i \in [1, q]$, has a fixed position with respect to $\mathcal{F}_\mathcal{W}$.

A configuration of an arbitrary object is a specification of the position of every point in this object relative to a fixed reference frame [Arnold, 1978]. Therefore, a **configuration** $\mathbf{q}$ of $\mathcal{A}$ is a specification of the position $\mathcal{T}$ and orientation Θ of $\mathcal{F}_\mathcal{A}$ with respect to $\mathcal{F}_\mathcal{W}$. The **configuration space** of $\mathcal{A}$ is the space $\mathcal{C}$ of all the configurations of $\mathcal{A}$. The subset of $\mathcal{W}$ occupied by $\mathcal{A}$ at configuration $\mathbf{q}$ is denoted by $\mathcal{A}(\mathbf{q})$. In the same fashion, the point a on $\mathcal{A}$ at configuration $\mathbf{q}$ is denoted by $a(\mathbf{q})$ in $\mathcal{W}$.

One may describe a configuration by a list of real parameters. For example, the position $\mathcal{T}$ can be simply described by the vector of the N coordinates of the origin of $\mathcal{F}_\mathcal{A}$ in $\mathcal{F}_\mathcal{W}$. The orientation Θ can be described as the $N \times N$ matrix whose columns are the components, in $\mathcal{F}_\mathcal{W}$, of the unit vectors along the axes of $\mathcal{F}_\mathcal{A}$. Then, $\mathbf{q} = (\mathcal{T}, \Theta)$ is uniquely represented as a list of $N(N+1)$ parameters. Clearly, however, this representation is redundant, since the matrix describing Θ must have orthonormal columns and determinant $+1$. Therefore, $\mathcal{C}$ is only a subset of $\mathbf{R}^{N(N+1)}$.

One may be tempted to describe Θ by a list of parameters of minimal cardinality, i.e. one angle θ, if $N = 2$ (for example, the angle between the x-axes of $\mathcal{F}_\mathcal{W}$ and $\mathcal{F}_\mathcal{A}$), and three angles ϕ, θ and ψ, if $N = 3$ (for example, the Euler angles). However, such a minimal description is not injective, i.e. the same configuration may be described by different values of the angles. For example, in the two-dimensional case ($N = 2$), two angles whose difference is a multiple of 2π represent the same Θ. In this case, the difficulty can easily be fixed by restricting the values of the angle θ to the $[0, 2\pi)$ interval with modulo 2π arithmetic. In the three-dimensional case ($N = 3$), the difficulty is slightly harder to remedy.

Describing a configuration as a list of m independent parameters, with

$m = 3$ (if $N = 2$) and $m = 6$ (if $N = 3$), corresponds to representing $\mathcal{C}$ as $\mathbf{R}^m$. However, as we will see at greater length in Chapter 2, while $\mathcal{C}$ is locally "like" $\mathbf{R}^m$, it is globally different. This means that $\mathcal{C}$ can be decomposed into a finite union of p ($p > 1$) slightly overlapping "patches", each represented as a copy of $\mathbf{R}^m$ (called a *chart*). Hence, a configuration can locally be represented as a non-redundant list of m real parameters, but some administration is required to switch between representations when the robot's configuration moves from one patch to another. The number m is called the **dimension** of $\mathcal{C}$.

3.2 Notion of a Path

The notion of *continuity* is fundamental in the definition of a path. Its formalization requires us to define a topology in $\mathcal{C}$. One classical way to do so is to specify a distance function $d : \mathcal{C} \times \mathcal{C} \rightarrow \mathbf{R}$, and to let the topology in $\mathcal{C}$ be the metric topology induced by d. In order to match our intuition of the real-world physics, the distance between two configurations $\mathbf{q}$ and $\mathbf{q}'$ should decrease and tend toward zero when the regions $\mathcal{A}(\mathbf{q})$ and $\mathcal{A}(\mathbf{q}')$ get closer to each other and tend to coincide. A simple distance function that satisfies this condition is:

$$d(\mathbf{q}, \mathbf{q}') = \max_{a \in \mathcal{A}} \|a(\mathbf{q}) - a(\mathbf{q}')\|$$

where $\|x - x'\|$ denotes the Euclidean distance between any two points x and x' in $\mathbf{R}^N$. Throughout this book, the topology in $\mathcal{C}$ will be the topology induced by this distance.[1]

A **path** of $\mathcal{A}$ from the configuration $\mathbf{q}_{init}$ to the configuration $\mathbf{q}_{goal}$ is a continuous map:

$$\tau : [0, 1] \rightarrow \mathcal{C}$$

with:

$$\tau(0) = \mathbf{q}_{init} \quad \text{and} \quad \tau(1) = \mathbf{q}_{goal}.$$

$\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ are the **initial** and **goal** configurations of the path, respectively. The continuity of τ entails that for all $s_0 \in [0, 1]$, $\lim_{s \rightarrow s_0} \max_{a \in \mathcal{A}} \|a(\tau(s)) - a(\tau(s_0))\| = 0$, with s taking its values in $[0, 1]$. Saying that $\mathcal{A}$ is a "free-flying object" means that, in the absence of obstacles, any path defined as above is feasible.

[1] As is well-known, the notion of a topology is more primitive than the notion of a metric. In Chapter 2 we will give other metrics which all induce the same topology as d.

3.3 Obstacles in Configuration Space

The above definition of a path does not take obstacles into consideration. We now want to characterize the set of paths which are solutions to the basic problem when there are obstacles in the workspace. To that purpose, we map the obstacles in configuration space and we consider the subset of $\mathcal{C}$ that is made of contact-free configurations.

Every obstacle $\mathcal{B}_i$, $i = 1$ to q, in the workspace $\mathcal{W}$ maps in $\mathcal{C}$ to a region:

$$\mathcal{CB}_i = \{\mathbf{q} \in \mathcal{C} \;/\; \mathcal{A}(\mathbf{q}) \cap \mathcal{B}_i \neq \emptyset\}$$

which is called a **C-obstacle**. The union of all the C-obstacles:

$$\bigcup_{i=1}^{q} \mathcal{CB}_i$$

is called the **C-obstacle region**, and the set:

$$\mathcal{C}_{free} = \mathcal{C} \setminus \bigcup_{i=1}^{q} \mathcal{CB}_i = \{\mathbf{q} \in \mathcal{C} \;/\; \mathcal{A}(\mathbf{q}) \cap (\bigcup_{i=1}^{q} \mathcal{B}_i) = \emptyset\}$$

is called the **free space**. Any configuration in $\mathcal{C}_{free}$ is called a **free configuration**.

A **free path** between two free configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ is a continuous map $\tau : [0,1] \rightarrow \mathcal{C}_{free}$, with $\tau(0) = \mathbf{q}_{init}$ and $\tau(1) = \mathbf{q}_{goal}$. Two configurations belong to the same connected component of $\mathcal{C}_{free}$ if and only if they are connected by a free path.

Given an initial and a goal configuration, the basic motion planning problem is to generate a free path between the two configurations, if they belong to the same connected component of $\mathcal{C}_{free}$, and to report failure otherwise.

Several planning methods described in Chapters 4 through 7 generate semi-free paths, rather than free paths. A **semi-free path** is a continuous map $\tau : [0,1] \rightarrow cl(\mathcal{C}_{free})$, where $cl(\mathcal{C}_{free})$ denotes the closure of $\mathcal{C}_{free}$. Hence, as it moves along such a path, the robot may touch obstacles. Under simple assumptions stated in Chapter 2, any semi-free path τ can be transformed into a free path that can be made arbitrarily close to τ.

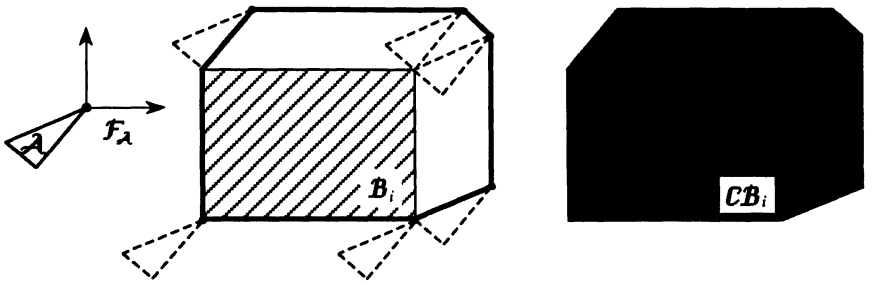

Figure 3. The robot $\mathcal{A}$ (a triangle) can translate freely in the plane at fixed orientation. Its configuration is represented as $\mathbf{q} = (x, y)$, the coordinates in $\mathcal{F}_\mathcal{W}$ of the vertex of $\mathcal{A}$ marked as a small circle (the origin of $\mathcal{F}_\mathcal{A}$). Hence, $\mathcal{A}$'s configuration space is $\mathcal{C} = \mathbf{R}^2$. The C-obstacle $\mathcal{CB}_i$ (shown dark) is obtained by "growing" the corresponding workspace obstacle $\mathcal{B}_i$ (a rectangle) by the shape of $\mathcal{A}$. Planning a motion of $\mathcal{A}$ relative to $\mathcal{B}_i$ is equivalent to planning a motion of the marked vertex of $\mathcal{A}$ relative to $\mathcal{CB}_i$.

3.4 Translational Case

When $\mathcal{A}$ consists of a single point (*point robot*), it is meaningless to talk about its orientation. Then, like $\mathcal{W}$, the configuration space of $\mathcal{A}$ is a copy of $\mathbf{R}^N$, and hence is a Euclidean space. The C-obstacles are identical to the obstacles in the workspace.

When $\mathcal{A}$ is a disc or when $\mathcal{A}$ is a dimensioned object allowed to translate freely (i.e. the origin of $\mathcal{F}_\mathcal{A}$ can follow any path in $\mathcal{W}$) without rotation (i.e. $\mathcal{F}_\mathcal{A}$ has a fixed orientation with respect to $\mathcal{F}_\mathcal{W}$), the configuration space $\mathcal{C}$ is also $\mathbf{R}^N$. But, in these cases, the C-obstacles are obtained by "growing" the obstacles by the shape of $\mathcal{A}$ as illustrated in Figure 3.

In the following, all these cases, where $\mathcal{C} = \mathbf{R}^N$, will be considered as particular cases of the basic motion planning problem. They will be convenient for introducing motion planning methods with simple examples.

4 Planning Approaches

There exists a large number of methods for solving the basic motion planning problem. Not all of them solve the problem in its full generality, however. For instance, some methods require the workspace to

be two-dimensional and the objects to be polygonal. Despite many external differences, the methods are based on few different general approaches: *roadmap*, *cell decomposition*, and *potential field*. We briefly introduce these approaches below and we illustrate them with simple examples in two-dimensional configuration spaces populated by polygonal C-obstacles. As shown in Figure 3, such configuration spaces correspond to the case where the robot $\mathcal{A}$ is a polygon (or a point) translating in a workspace containing polygonal obstacles. These approaches will be described in detail in Chapters 4 through 7.

4.1 Roadmap

The roadmap approach to path planning consists of capturing the connectivity of the robot's free space in a network of one-dimensional curves, called the *roadmap*, lying in the free space $\mathcal{C}_{free}$ or its closure $cl(\mathcal{C}_{free})$. Once a roadmap $\mathcal{R}$ has been constructed, it is used as a set of standardized paths. Path planning is thus reduced to connecting the initial and goal configurations to points in $\mathcal{R}$ and searching $\mathcal{R}$ for a path between these points. The constructed path, if any, is the concatenation of three subpaths: a subpath connecting the initial configuration to the roadmap, a subpath contained in the roadmap, and a subpath connecting the roadmap to the goal configuration.

Various methods based on this general idea have been proposed. They compute different types of roadmaps called *visibility graph*, *Voronoi diagram*, *freeway net* and *silhouette*. We illustrate two of them below.

The *visibility graph method* is one of the earliest path planning methods. It mainly applies to two-dimensional configuration spaces with a polygonal C-obstacle region. The visibility graph is the non-directed graph G whose nodes are the initial and goal configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, and all the C-obstacle vertices. The links of G are all the straight line segments connecting two nodes that do not intersect the interior of the C-obstacle region. Figure 4 shows an example of a visibility graph. The links of the subgraph of G that is restricted to the C-obstacle vertices determine the roadmap $\mathcal{R}$. The other links of G connect the initial and goal configurations to the roadmap. G can be searched for the shortest semi-free path between $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ (according to the Euclidean metric in $\mathbf{R}^2$). This path, if it exists, is a polygonal line connecting $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ through C-obstacle vertices.

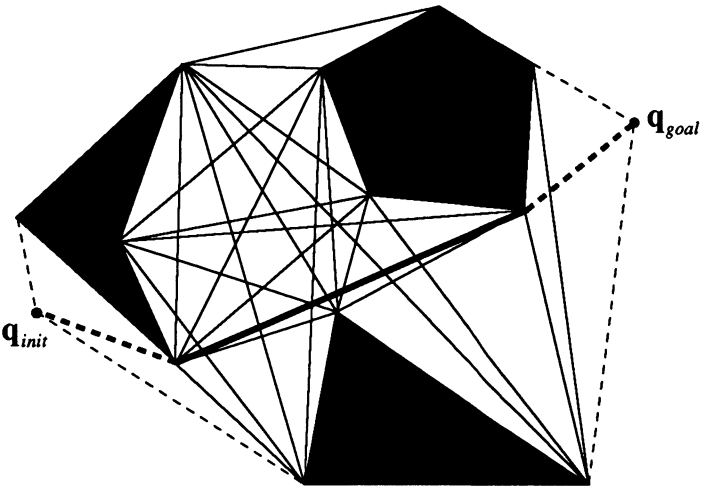

Figure 4. This figure shows the visibility graph in a two-dimensional configuration space with polygonal C-obstacles. The nodes of the graph are the initial and goal configurations and the C-obstacles' vertices. Two nodes are connected by a link if the straight line segment joining them does not intersect the C-obstacles' interior. (Hence, the links of the graph also include the edges of the C-obstacles.) The plain links connect C-obstacles' vertices. They are independent of the initial and goal configurations, and form the roadmap. The dashed links connect the initial and goal configurations to the roadmap. Any path in the visibility graph determines a semi-free path in configuration space. The graph contains the shortest semi-free path (according to the Euclidean metric in $\mathbf{R}^2$). This path is shown in both plain and dashed bold lines.

Another roadmap method, called *retraction*, consists of defining a continuous function of $\mathcal{C}_{free}$ onto a one-dimensional subset of itself (the roadmap) such that the restriction of this function to this subset is the identity map (in Topology such a function is called a "retraction"). In a two-dimensional configuration space, $\mathcal{C}_{free}$ is typically "retracted" onto its *Voronoi diagram*. This diagram is the set of all the free configurations whose minimal distance to the C-obstacle region $\mathcal{CB}$ is achieved with at least two points in the boundary of $\mathcal{CB}$ (Figure 5). The advantage of this diagram is that it yields free paths which tend to maximize the clearance between the robot and the obstacles. When the C-obstacles are polygons, the Voronoi diagram consists of straight and parabolic segments.

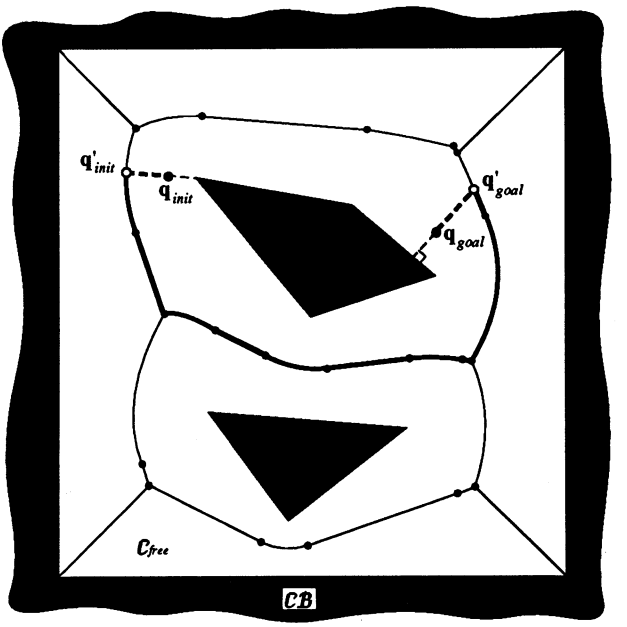

Figure 5. This figure shows the Voronoi diagram in a configuration space with a polygonal C-obstacle region. The free space in externally bounded by a rectangle (the boundary of a non-compact C-obstacle). The initial and goal configurations, $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, are mapped in the Voronoi diagram to $\mathbf{q}'_{init}$ and $\mathbf{q}'_{goal}$, each by drawing the line along which its distance to the boundary of the C-obstacle region increases the fastest.

The initial and goal configurations are retracted in the diagram to $\mathbf{q}'_{init}$ and $\mathbf{q}'_{goal}$ as shown in Figure 5. A path is searched in the diagram between $\mathbf{q}'_{init}$ and $\mathbf{q}'_{goal}$. The free path between $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ produced by the method, if any, consists of three subpaths: the straight path from $\mathbf{q}_{init}$ to $\mathbf{q}'_{init}$, a path in the diagram from $\mathbf{q}'_{init}$ to $\mathbf{q}'_{goal}$, and the straight path from $\mathbf{q}'_{goal}$ to $\mathbf{q}_{goal}$.

4.2 Cell Decomposition

Cell decomposition methods are perhaps the motion planning methods which have been the most extensively studied so far. They consist of decomposing the robot's free space into simple regions, called *cells*, such that a path between any two configurations in a cell can be easily gener-

ated. A non-directed graph representing the adjacency relation between the cells is then constructed and searched. This graph is called the *connectivity graph.* Its nodes are the cells extracted from the free space and two nodes are connected by a link if and only if the two corresponding cells are adjacent. The outcome of the search is a sequence of cells called a *channel.* A continuous free path can be computed from this sequence.

Cell decomposition methods can be broken down further into *exact* and *approximate* methods:

- Exact cell decomposition methods decompose the free space into cells whose union is exactly[2] the free space. The boundary of a cell corresponds to a criticality of some sort, i.e. a sudden change in the constraints applying to the motion of the robot.

- Approximate cell decomposition methods produce cells of predefined shape (e.g. rectangloids) whose union is strictly included in the free space. The boundary of a cell does not characterize a discontinuity of some sort and has no physical meaning.

Figures 6.1 and 6.2 illustrate an exact cell decomposition method in a two-dimensional configuration space. The free space is externally bounded by a polygon and internally bounded by three polygons. The free space is exactly decomposed into trapezoidal and triangular cells. These cells are built by drawing vertical rays from the C-obstacles' vertices. Two cells are adjacent if they share a portion of an edge of non-zero length. Once the connectivity graph has been constructed and a channel found, a free path is computed by connecting the initial configuration to the goal configuration through the midpoints of the intersections of every two successive cells.

Figure 7 illustrates an approximate cell decomposition method. The free space is externally bounded by a rectangle R and internally bounded by three polygons. The rectangle R is recursively decomposed into smaller rectangles. Each decomposition generates four identical new rectangles. (This type of decomposition is called a "quadtree" because it can be represented as a tree of degree 4.) At some level of resolution, only the

[2]The term "exact" refers to the input description of the motion planning problem, not to the actual physical problem, which can only be approximated in the input problem formulation.

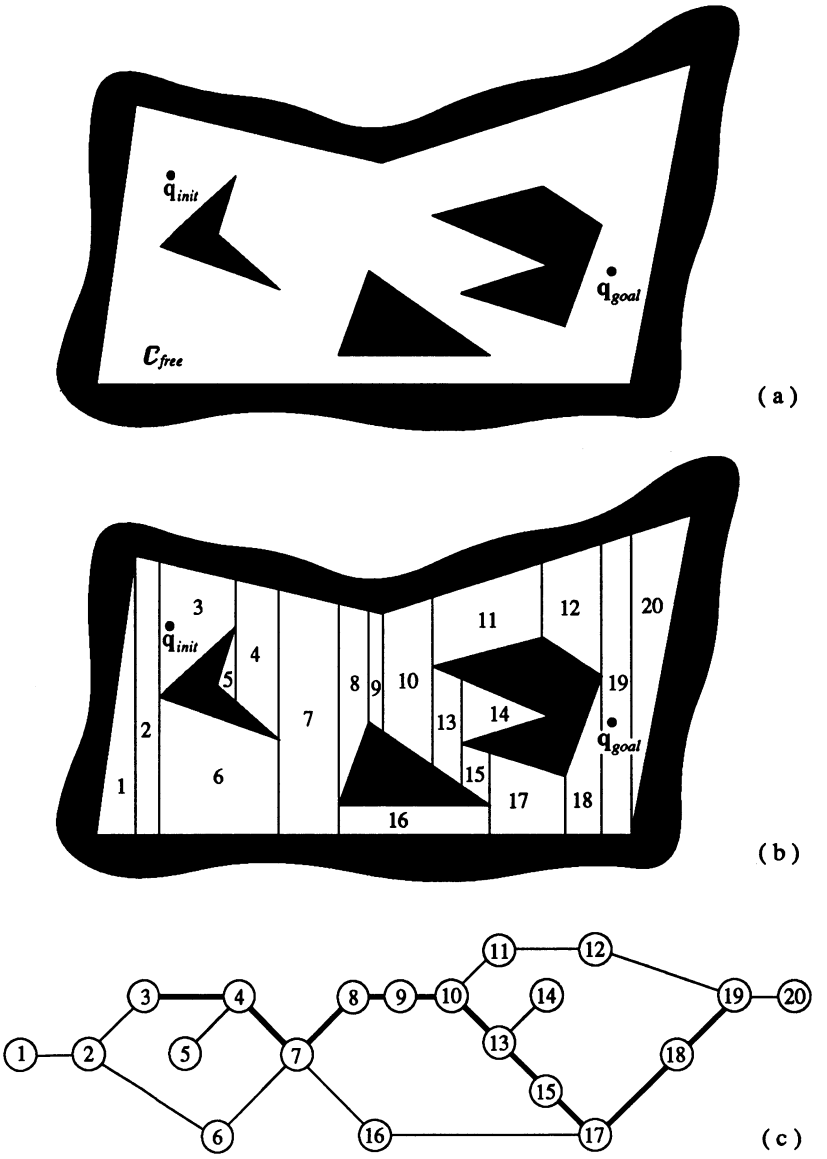

Figure 6.1. This figure and the next one illustrate an exact cell decomposition method. The free space is externally bounded by a polygon and internally bounded by three polygons (Figure a). The first step of the method consists of decomposing the free space into trapezoidal and triangular cells (Figure b). The second step is to construct the connectivity graph representing the adjacency relation between the cells (Figure c) and search this graph for a path (thick lines).

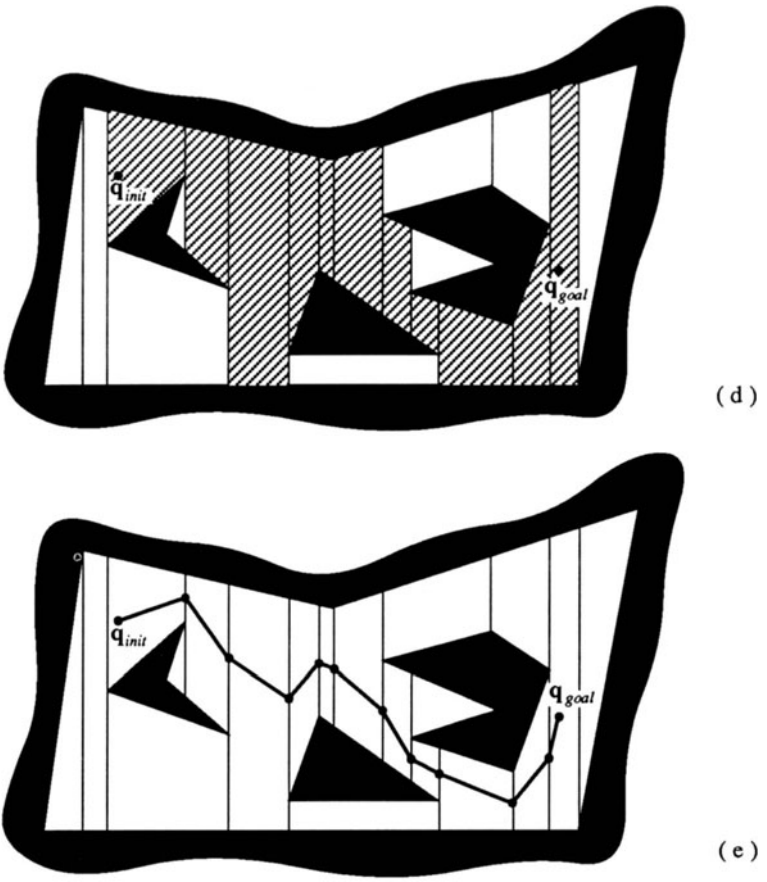

Figure 6.2. A path in the connectivity graph determines a channel (the sequence of striped cells in Figure d) in free space. The third step of the method consists of transforming this channel into a free path, for example by connecting the initial configuration to the goal configuration through the midpoints of the intersections of every two successive cells (Figure e).

cells whose interiors lie entirely in the free space are used to construct the connectivity graph. If the search of this graph terminates successfully, a path in the free space is easily generated. Otherwise, it may mean that either the resolution of the decomposition is insufficient, or no free path exists between the initial and goal configurations. Often, an approximate cell decomposition method operates in a hierarchical fashion by using a

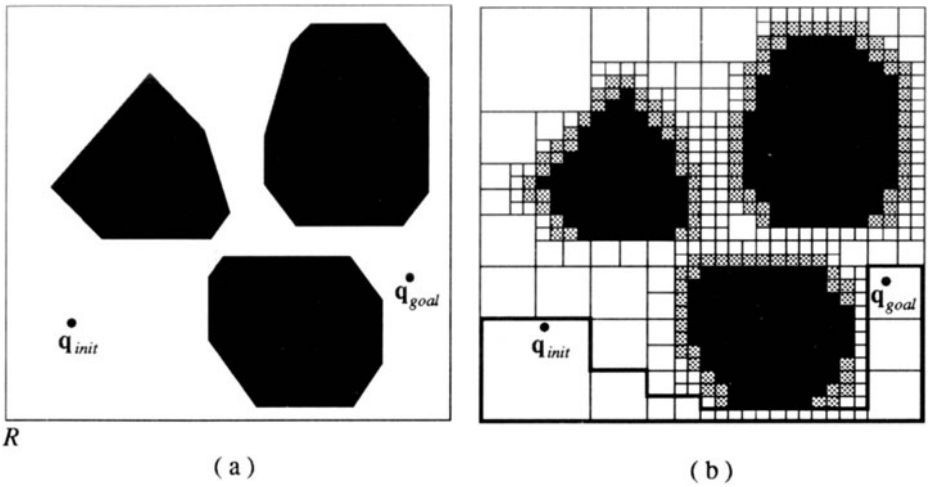

Figure 7. This figure illustrates an approximate cell decomposition method. The free space is externally bounded by a rectangle R and internally bounded by three polygons (Figure a). The rectangle R is decomposed into four identical rectangles. If the interior of a rectangle lies completely in free space or in the C-obstacle region, it is not decomposed further. Otherwise, it is recursively decomposed into four rectangles until some predefined resolution is attained. The result is shown in Figure b. White cells lie outside the C-obstacle region; black cells lie inside the C-obstacle region; grey cells are cut by the C-obstacle region. This kind of decomposition is called a "quadtree" decomposition. A channel extracted from this decomposition is shown in bold contour.

rather coarse resolution at the beginning, and refining it until either a free path is found or a limit resolution is attained.

Provided that they are equipped with both appropriate search techniques and exact numerical computation techniques, exact cell decomposition methods are complete — i.e. they are guaranteed to find a free path whenever one exists and to return failure otherwise. Approximate methods may not be complete; but, for most of them, the precision of the approximation can be tuned and made arbitrarily small (at the expense of the running time), so that the methods are said to be "resolution-complete". On the other hand, except in very simple cases, exact methods are mathematically more involved than approximate ones. Hence, the latter are usually easier to implement.

4.3 Potential Field

A straightforward approach to motion planning is to discretize the configuration space into a fine regular grid of configurations and to search this grid for a free path. This approach requires powerful heuristics to guide the search, since the grid is in general enormous. Several types of heuristics have been proposed. The most successful ones take the form of functions that are interpreted as *potential fields.*

The metaphor suggested by this terminology is that the robot represented as a point in configuration space is a particle moving under the influence of an artificial potential produced by the goal configuration and the C-obstacles. Typically the goal configuration generates an "attractive potential" which pulls the robot toward the goal, and the C-obstacles produce a "repulsive potential" which pushes the robot away from them. The negated gradient of the total potential is treated as an artificial force applied to the robot. At every configuration, the direction of this force is considered the most promising direction of motion.

Figure 8 illustrates the notion of attractive and repulsive potentials, and their combination. The attractive potential (Figure 8.b) is a quadratic well with its minimum at the goal configuration. The repulsive potential (Figure 8.c) is non-zero only within some distance from the C-obstacles and tends toward infinity when the distance to the C-obstacles tends toward zero (hence, is truncated in the figure). A path between the initial and goal configurations is constructed by tracking the negated gradient of the total potential (Figure 8.e). A matrix of orientations of the negated gradient vector field (orientations of the artifical forces induced by the potential field) is displayed in Figure 8.f.

In comparison to other methods, potential field methods can be very efficient. However, they have a major drawback. Since they are essentially fastest descent optimization methods, they can get trapped into local minima of the potential function other than the goal configuration. One attack of this problem is to design potential functions that have no local minima other than the goal configuration in the connected subset of free space which contains the goal configuration. Another approach is to complement the basic potential field approach with powerful mechanisms to escape from local minima.

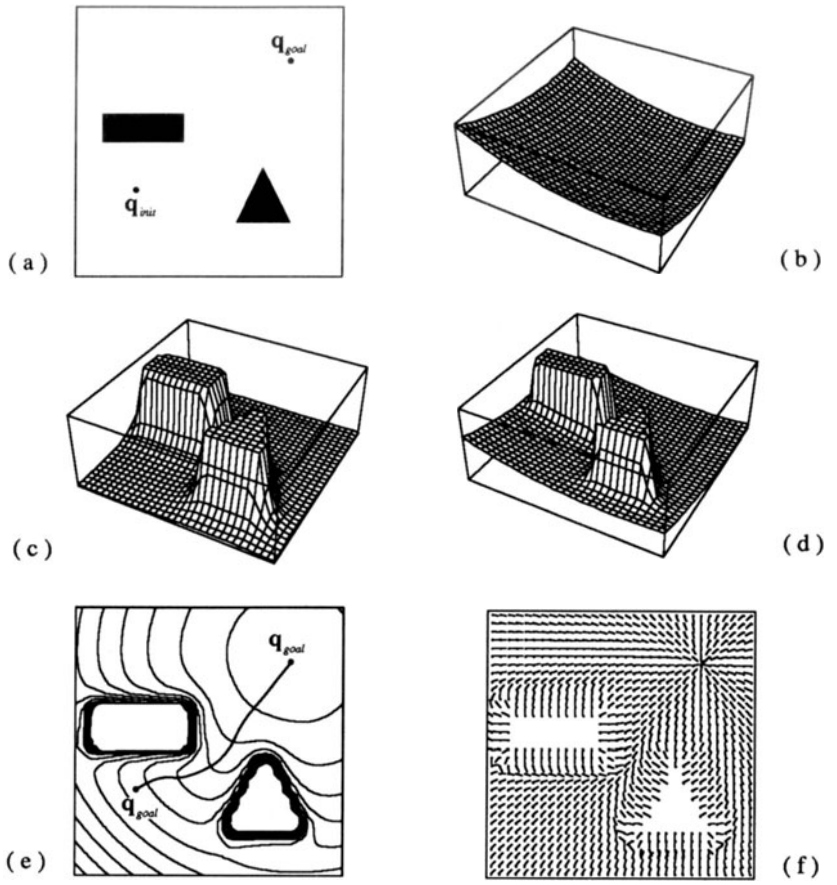

Figure 8. This series of diagrams illustrates the potential field approach. A simple two-dimensional configuration space with two polygonal C-obstacles is depicted in Figure a. Figure b shows an attractive potential generated by the goal configuration $\mathbf{q}_{goal}$. Figure c shows a repulsive potential generated by the C-obstacles. Figure d shows the sum of the two potentials. Figure e displays equipotential contours of the total potential and a path obtained by following its negated gradient. Figure f shows orientations of the negated gradient of the total potential over free space.

4.4 Global versus Local Methods

The roadmap and cell decomposition methods reduce the problem of finding a continuous free path to that of searching a graph (e.g. the

visibility graph, the Voronoi diagram, the connectivity graph) by first analyzing the connectivity of the free space. Unlike these methods, potential field methods do not include an initial processing step aimed at capturing the connectivity of the free space in a concise representation. Instead, they search a much larger graph representing the adjacency among the configurations contained in the grid thrown over configuration space. At each step, they move from one configuration in the grid to another one, basing their choice on the computation of the potential gradient. It is typical for this gradient to depend only on the contents of the configuration space in the neighborhood of the current configuration of the robot. (As illustrated in Figure 8.c, the repulsive potential is usually defined with a limited range of influence around the C-obstacles.) Therefore, potential field methods are often called *local* methods, while the roadmap and cell decomposition methods are called *global* methods.

In practice, this distinction is certainly valid. For example, a potential field method can be run while the robot is moving, by tracking the negated gradient of a potential field defined as a function of both the distance to the goal configuration and the distances to the obstacles as measured by range sensors. The distinction is nevertheless intuitive and has no solid theoretical basis. A potential field method may use a potential function that is free of local minima (in the connected subset containing the goal configuration). Computing such a function certainly requires the geometry of the whole free space to be taken into account. Then the method is as global as any roadmap or cell decomposition method. On the other hand, a roadmap or cell decomposition method can be made local by restricting its application to a subset of the configuration space around the current configuration of the robot. A path is then generated in an iterative fashion by concatenating subpaths, each achieving an intermediate configuration in such a restricted subset.

5 Extensions of the Basic Problem

The basic motion planning problem makes assumptions that significantly limit the practicality of its solutions. In other words, it may be quite difficult to reduce an actual robotic problem to an instance of the basic problem, to solve this instance, and to adapt the produced solution in order to match the conditions of the original problem. In this section we

introduce and discuss informally several important extensions that we will investigate in detail in Chapters 8 through 11. Other extensions can certainly be envisioned, but they have received too little attention so far to be treated in this book.

5.1 Multiple Moving Objects

In the basic problem we assumed that the obstacles were fixed, that there was a single robot, and that this robot was made of a single rigid object. Under the title "multiple moving objects", we consider a series of extensions that removes these assumptions. One extension consists of including *moving obstacles* in the workspace. The second one allows *multiple robots* to operate in the same workspace, each being a potential obstacle to the others. The third extension considers *articulated robots*, which are made of several rigid objects connected by joints.

The first extension (and possibly the second) requires time to be explicitly considered in the motion plans. The second and third extensions yield configuration spaces of arbitrarily large dimensions.[3]

Moving Obstacle. In the presence of moving obstacles, the motion planning problem can no longer be solved by merely constructing a geometric path. A continuous function of time specifying the robot's configuration at each instant must be generated instead. This can be done by adding one dimension representing time to the robot's configuration space. The new space, denoted by $\mathcal{CT}$, is called *configuration-time space*[4]. The workspace obstacles map in $\mathcal{CT}$ to static regions called *CT-obstacles*. Every cross-section through $\mathcal{CT}$ at time t is the configuration space of the robot at instant t. It cuts the CT-obstacles according to the C-obstacles corresponding to the workspace obstacles at their locations at instant t. Motion planning consists of finding a path among the CT-obstacles in $\mathcal{CT}$. Since time is irreversible, this path must have the property of never going backward along the time axis. Hence, path planning methods have to be modified in order to take this specificity of the time dimension into account.

[3] In Section 6 we will see that the inherent computational complexity of path planning is exponential in the dimension of the robot's configuration space.

[4] In the literature it is also called configuration space-time.

If there is no constraint on the robot's velocity and acceleration, and if the motion of every obstacle is known beforehand, it is rather straightforward to extend some basic planning methods to handle this new problem. If constraints apply to the robot's velocity and/or its acceleration, they translate to geometric constraints on the slope and the curvature of a path along the time dimension, and the problem turns out to be significantly harder.

Multiple Robots. Motion planning with multiple robots differs from planning in the presence of moving obstacles in that the motions of the robots have to be planned, while the motions of the moving obstacles are not under control.

One way to deal with multiple robots $\mathcal{A}_1, ..., \mathcal{A}_p$ operating in the same workspace $\mathcal{W}$ is to treat them as a single multi-bodied robot $\mathcal{A} = \{\mathcal{A}_1, ..., \mathcal{A}_p\}$. The *composite configuration space* of $\mathcal{A}$ is $\mathcal{C} = \mathcal{C}_1 \times ... \times \mathcal{C}_p$, i.e. the set product of the configuration spaces $\mathcal{C}_i$ of the individual robots $\mathcal{A}_1, ..., \mathcal{A}_p$. Every configuration in $\mathcal{C}$ determines a unique position and orientation for each robot $\mathcal{A}_i$. C-obstacles in this space result from the interaction of either a robot $\mathcal{A}_i$ and an obstacle $\mathcal{B}_j$, or two robots $\mathcal{A}_i$ and $\mathcal{A}_j$. Basic path planning methods can be used to plan a path of $\mathcal{A}$ in $\mathcal{C}$. This approach to multi-robot path planning is called *centralized planning*. A difficulty is that it may yield high-dimensional configuration spaces. Indeed, the dimension of the composite configuration space $\mathcal{C}$ is equal to the sum of the dimensions of the individual configuration spaces $\mathcal{C}_1$ through $\mathcal{C}_p$.

Another approach to motion planning with multiple robots, *decoupled planning*, is to plan the motion of each robot more or less independently of the other robots and to consider the interactions among the paths in a second phase of planning. By doing so, the amount of computation may be significantly reduced, but at the expense of completeness. Figure 9 illustrates a planning problem where the decoupled approach is likely to fail. The problem consists of interchanging the positions of two robots (discs) in a narrow corridor where they cannot pass each other. There is enough room at one end of the corridor for a permutation of the two robots. However, by considering each robot separately, the decoupled planning approach has no mechanism to infer that both robots must first move to this end of the corridor.

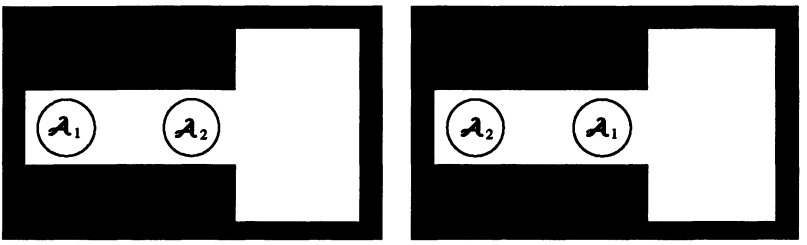

Figure 9. Two discs $\mathcal{A}_1$ and $\mathcal{A}_2$ have to interchange their positions in a narrow corridor where they cannot pass each other. However, there is enough room for permutation at one end of the corridor. Decoupled planning, which considers the two robots separately, would very likely fail to solve this problem.

Articulated Robots. An articulated robot $\mathcal{A}$ is made of several moving rigid objects $\mathcal{A}_1, ..., \mathcal{A}_p$ (called *links*) connected by *joints*, e.g. revolute joints (hinges) and prismatic joints (sliding joints). Each joint constrains the relative movements of the two objects it connects. A typical example of an articulated robot is a robot arm (see Figure 10).

$\mathcal{A}$ can be seen as a set of p moving rigid objects. The constraints imposed by the joints on the relative movements of the $\mathcal{A}_i$'s determine a subset of the composite configuration space of these objects, which is the actual configuration space of $\mathcal{A}$. Every configuration in this subset indeed determines a unique position and orientation for each of the p links. This subset is in general relatively easy to parameterize, for example by associating a distance or an angle with each joint. Several basic motion planning methods extend to this case in a straightforward fashion, at least in theory. A practical problem that has to be faced is that the dimension of the configuration space grows with the number of joints.

Figure 11 illustrates several configurations of a planar articulated robot along a path generated by a potential field method. In this example, the robot is a sequence of eight links. All the joints are revolute joints. Notice that for an articulated robot with many links and joints, it may be problematic to specify a free goal configuration. In the example of Figure 11, only the goal position of the endpoint of the arm was specified to the planner. This corresponds to defining a goal region in the robot's configuration space, rather than a unique goal configuration.

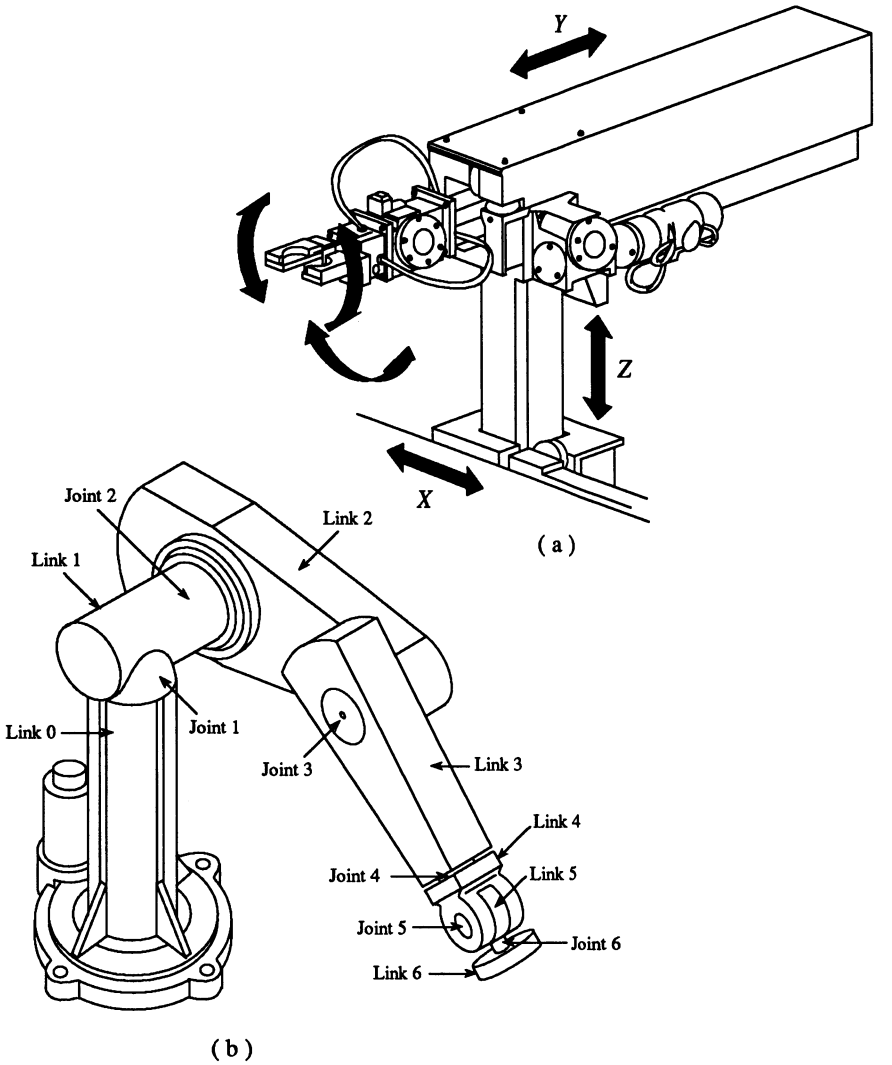

Figure 10. A robot arm is a typical example of an articulated robot. It is made of several rigid objects, called links, serially connected by joints. Most robot arms have revolute joints and/or prismatic joints. A revolute joint constrains the relative motion of two objects to be a rotation around an axis fixed with respect to both objects. A prismatic joint constrains the relative motion of two objects to be a translation along an axis fixed with respect to both objects. Figure a shows a Cartesian robot, with three prismatic joints and three revolute ones. Figure b shows a robot with six revolute joints.

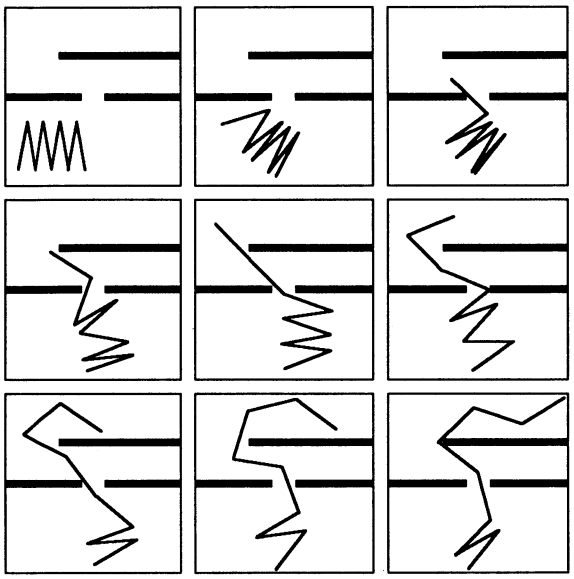

Figure 11. This figure shows a sequence of configurations of a planar eight-revolute-joint arm along a path generated using a potential field method.

5.2 Kinematic Constraints

In the basic problem we assumed that the robot was a free-flying object, i.e. the only constraints on its motions were due to the obstacles. In some problems we may want to impose additional kinematic constraints to the robot's motions. Two types of constraints, called *holonomic constraints* and *nonholonomic constraints*, can be considered.

Holonomic Constraints. Let us assume that a configuration is represented by a list of parameters of minimal cardinality (see Section 3). A holonomic equality constraint is an equality relation among these parameters, which can be solved for one of the parameters. Such a relation reduces the dimension of the actual configuration space of the robot by one. A set of k independent holonomic constraints reduces it by k.

Consider for example the case of a three-dimensional object $\mathcal{A}$ allowed to translate freely, but constrained to rotate around a fixed axis relative to $\mathcal{A}$. Let us represent the orientation of $\mathcal{A}$ by three angles. The

constraints restricting the possible orientations of $\mathcal{A}$ can be expressed as two independent equations among these angles. While the dimension of the configuration space of a free-flying object in a three-dimensional workspace is 6, $\mathcal{A}$'s configuration space has dimension 4. The particular case where $\mathcal{A}$ can translate freely at fixed orientation could also be considered as a problem with holonomic constraints. However, since this problem is equivalent to the problem of planning the motion of a point robot in $\mathbf{R}^N$, we regard it as a particular case of the basic motion planning problem.

Articulated robots provide another example of holonomic constraints. For instance, a revolute joint determines two holonomic constraints. Indeed, while six parameters are necessary to define the configuration of two free-flying planar objects, four parameters suffice to determine the positions and orientations of two planar objects connected by a revolute joint. As another example, consider a six-joint arm carrying a glass of water. Requiring the glass to stay vertical during the motion corresponds to imposing two holonomic constraints to the arm.

Holonomic constraints certainly affect the definition of the robot's configuration space and may even change its global connectedness. Nonetheless, holonomic constraints do not raise new fundamental issues. Many basic planning methods remain applicable.

Nonholonomic Constraints. A nonholonomic equality constraint is a non-integrable equation involving the configuration parameters and their derivatives (velocity parameters). Such a constraint does not reduce the dimension of the space of configurations attainable by the robot, but reduces the dimension of the space of possible differential motions (i.e. the space of the velocity directions) at any given configuration.

Consider for example a car-like robot $\mathcal{A}$ rolling on a flat ground. We model it as a rectangular object moving in $\mathcal{W} = \mathbf{R}^2$ (see Figure 12). We know by experience that in an empty space we can drive the robot to any position with any orientation. Hence the robot's configuration space has three dimensions, two of translation and one of rotation. Let us represent a configuration of $\mathcal{A}$ by (x, y, θ), where x and y are the coordinates, in the frame $\mathcal{F}_\mathcal{W}$, of the midpoint R between the two rear wheels and $\theta \in [0, 2\pi)$ is the angle between the x-axis of $\mathcal{F}_\mathcal{W}$ and the main axis of $\mathcal{A}$. At any instant during a motion, assuming no slipping,

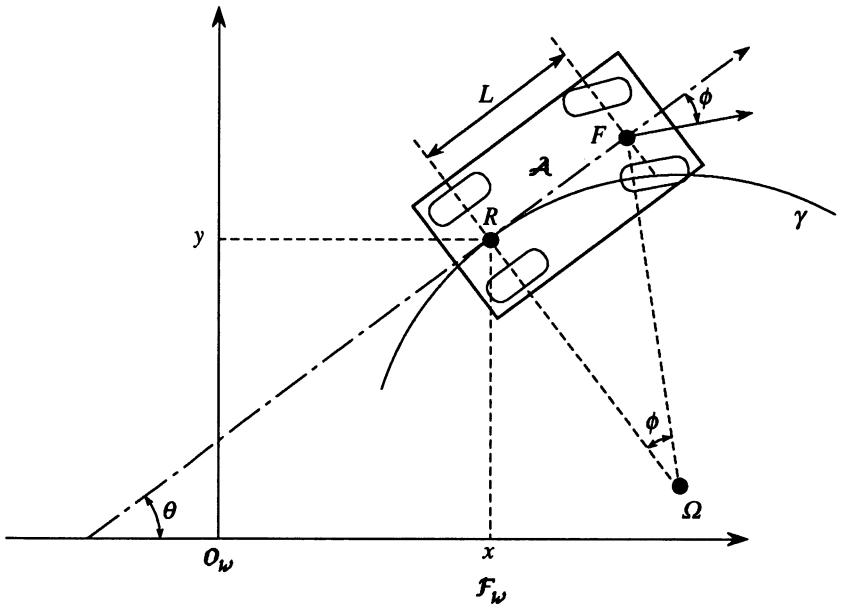

Figure 12. This car-like robot $\mathcal{A}$ is modeled as a rectangle. Any configuration of $\mathcal{A}$ is represented by three parameters $(x, y, \theta) \in \mathbf{R}^2 \times [0, 2\pi)$, with modulo 2π arithmetic on θ. Although in the absence of obstacles the robot can attain any configuration in $\mathbf{R}^2 \times [0, 2\pi)$, the projection of any path of $\mathcal{A}$ on the xy-plane is a curve γ tangent to the main axis of $\mathcal{A}$, thus implying that the relation $-\sin\theta\, dx + \cos\theta\, dy = 0$ be verified. This relation is a nonholonomic equality constraint. The curvature of γ is $L/\tan\phi$, where L is the distance between R and F, the midpoint between the two front wheels, and ϕ is the steering angle, i.e. the angle between $\mathcal{A}$'s main axis and the velocity vector of point F. In general, $|\phi| \leq \phi_{max} < \pi/2$.

the velocity of R has to point along the main axis of $\mathcal{A}$. Therefore, the motion is constrained by the relation:

$$-\sin\theta\, dx + \cos\theta\, dy = 0.$$

It can be shown that this equation is non-integrable, hence is a nonholonomic equality constraint. Due to this constraint, the space of differential motions $(dx, dy, d\theta)$ of the robot at any configuration (x, y, θ) is a two-dimensional space. If the robot was a free-flying object, this space would be three-dimensional. The instantaneous motion of the car-like robot is

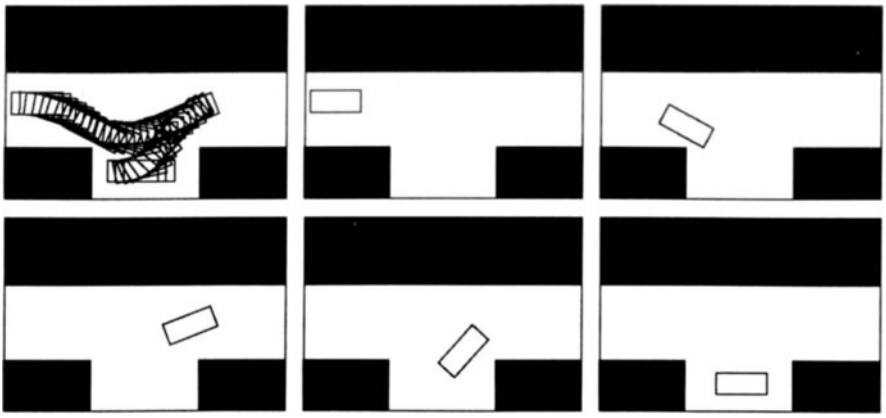

Figure 13. This figure illustrates a path generated by a planner described in Chapter 9 for a car-like robot in order to perform a parallel parking operation.

determined by two parameters: the linear velocity along its main axis and the steering angle. However, when the steering angle is non-zero, the robot changes orientation, and its linear velocity with it, allowing the robot's configurations to span a three-dimensional space.

Furthermore, the steering angle denoted by ϕ in Figure 12 is in general constrained to take values in an interval $[-\phi_{max}, +\phi_{max}]$, with $\phi_{max} < \pi/2$. This constraint can be rewritten as an inequality involving dx, dy and $d\theta$. It restricts the set of possible differential motions without changing its dimension. We call it a nonholonomic inequality constraint.

Nonholonomic constraints restrict the geometry of the feasible free paths between two configurations. They are much harder to deal with in a planner than holonomic constraints. Figure 13 illustrates a path generated for a car-like robot in order to perform a parallel parking operation.

5.3 Uncertainty

The basic problem assumes that the robot can follow the paths generated by the planner exactly. It also assumes that the geometry of the robot, the geometry of the obstacles, and the location of the obstacles are accurately known. No robot setting rigorously satisfies these assumptions, and both robot control and geometric models are imperfect. Fortunately,

we often do not have to worry about these imperfections because they are small relative to the tolerance of the task being performed. But this is not always the case.

At the other extreme, the robot could have little or no prior knowledge about its workspace. Then, it would have to rely on its sensors at execution time to obtain the information needed to accomplish the task. This extreme case requires the robot to explore its workspace, and it tends to be outside the realm of motion planning. The robot may interweave planning and execution monitoring activities, but the more incomplete the prior knowledge, the less important the role of planning.

An intermediate situation is when there are errors in robot control and in the initial geometric models, but these errors are contained within bounded regions. For example, the actual obstacles' locations are slightly different from those in the robot's model, but the errors on the parameters defining these locations are bounded. Similarly, the robot moves along a direction that is different from the commanded one, but the actual direction of motion is contained in a narrow cone centered along the commanded direction. In order to deal with such bounded errors, we assume that the robot is equipped with sensors that can be used at execution time to acquire additional information. But sensors are not perfect, either. For example, a position sensor does not return the exact configuration of the robot. Again, it is realistic to assume that the sensing errors are contained in bounded uncertainty regions. When errors in control, sensing and model are reasonably small, it is interesting to generate motion plans that are tolerant to these errors, i.e. that achieve goals reliably by anticipating the various possible contingencies.

The motion planning problem with bounded uncertainty can be stated as follows: Given an *initial region* $\mathcal{I}$ and a *goal region* $\mathcal{G}$ in the robot's configuration space, generate a motion plan whose execution guarantees the robot to reach a configuration in $\mathcal{G}$ if it starts from any (unknown) configuration in $\mathcal{I}$, despite bounded uncertainty in control, sensing and model. A solution to this problem is a plan that combines motion commands and sensor readings that interact at execution time in order to reduce uncertainty and guide the robot toward the goal.

Planning in the presence of bounded uncertainty raises new issues explored neither in the basic problem, nor in the previous extensions. Due

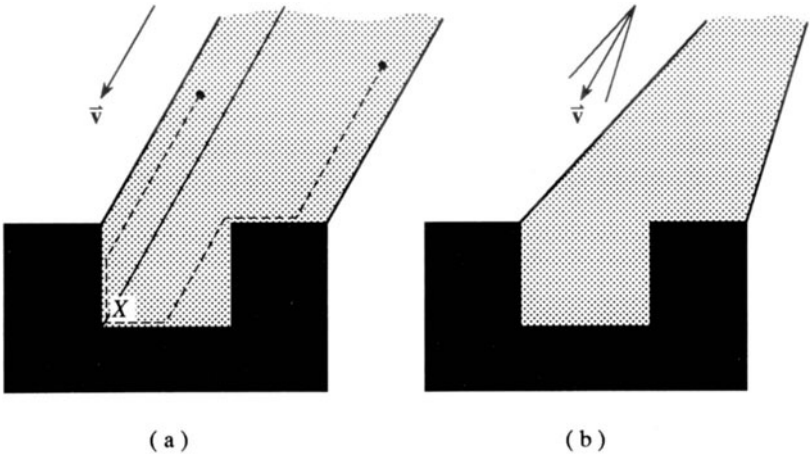

Figure 14. This figure illustrates a force-compliant motion in a two-dimensional configuration space with a polygonal C-obstacle. The robot is commanded to move along the direction $\vec{v}$. Assuming perfect control (Figure a), the robot moves along a straight line parallel to $\vec{v}$ as long as it is in free space. When it touches an obstacle, the robot switches to a path following the projection of $\vec{v}$ in the C-obstacle's boundary. The grey region in Figure a shows the range of initial configurations from which any compliant motion commanded along $\vec{v}$ reaches the vertex X. The two dashed lines illustrate two such motions. (The range of initial configurations from which a pure position-controlled motion commanded along $\vec{v}$ reaches X is the half-line parallel to $\vec{v}$ ending at X). If control is imperfect, e.g. the robot moves along a direction contained in a cone centered along the commanded direction $\vec{v}$ (see Figure b), the grey region is slightly reduced. (In contrast, the range of initial configurations from which a pure position-controlled motion commanded along $\vec{v}$ is guaranteed to reach X vanishes to the point X itself.)

to uncertainty in control, a motion command may produce any path among the infinitely many ones which are consistent with both the command and the uncertainty. In order for the planner to guarantee that execution will succeed, all of these paths must reach the goal (goal reachability). But this is only a sufficient condition. The plan must also terminate *in* the goal. Due to uncertainty in sensing, it may not be a trivial issue to recognize goal achievement reliably. The planner must make sure that enough information will be made available to the controller

during execution (goal recognizability).

Uncertainty often leads to using sensor-based motion commands whose behavior is less sensitive to errors than purely position-controlled motion commands. *Force-compliant* motion commands are one example of such commands. When they are used, the robot may touch obstacle surfaces and slide on them (along the projection of the commanded direction of motion), rather than just stop. Figure 14 illustrates a compliant motion of a robot commanded along the direction $\vec{v}$ in a two-dimensional configuration space. In Figure 14.a we assume perfect control; in Figure 14.b we assume imperfect control (the actual direction of motion is contained in a cone centered along $\vec{v}$). In both cases, the grey region is the set of initial configurations from where the robot commanded along $\vec{v}$ is guaranteed to reach the C-obstacle vertex X (assuming frictionless edges). The grey region in Figure 14.b is only slightly smaller than in Figure 14.a. In contrast, with uncertainty in control, no position-controlled motion commanded along $\vec{v}$ is guaranteed to reach X. Planning such sensory-based motion commands may require the physics of the workspace to be taken into consideration. For example, when force sensing is used to detect contact with the obstacles and slide in their boundary, frictional effects ignored in Figure 14 may be critical.

5.4 Movable Objects

In the basic problem the objects in the workspace other than the robot are all stationary, non-movable obstacles. In one extension introduced in Subsection 5.1 the obstacles can move, but their motions are not under the robot's control. Another important extension is when the workspace contains *movable objects*, i.e. objects that the robot can displace, for example by grasping them.

The presence of movable objects requires the generation of sequences of motion commands, each intended to achieve some intermediate goal which may not have been specified in the input problem. For example, even if there exists no feasible path to a goal configuration in some given arrangement of the workspace, the robot may create one by moving movable objects to other locations. In order to do so, the planner must select these other locations (intermediate goals). Hence, a motion plan appears as an alternate sequence of *transit* and *transfer* motions. Along a transit motion the robot moves alone. Along a transfer motion it

moves together with a movable object. The transit and transfer motions lie in different spaces. The transit motions occur in the configuration space of the robot, but different arrangements of the movable objects yield different geometries in this space. A transfer motion occurs in the configuration space of the union of the robot and the movable object moving with it. The main problem for the planner is to determine which spaces to use and when to change from one space to another.

The most common way for a robot to displace a movable object is to rigidly grasp it and to move with it. For example, if the robot is equipped with a multi-fingered gripper, grasping an object consists of placing the fingers of the gripper on the surface bounding the object and exerting some gripping force. Then, if the robot moves, the object moves with it as a single object. Grasping requires that the robot be capable of selecting the positions of the fingers on the object (another intermediate goal). Several constraints — force-closure, stability, tolerance to positioning errors, minimization of gripping forces, geometric feasibility — have to be satisfied by these positions. The grasp planning problem (a difficult problem in its own) is part of the motion planning problem in the presence of movable objects.

Grasping is not the only way of moving a movable object. The robot may also push an object on a table. It may also cause an object to move by tilting the surface supporting the object with respect to gravity. If it is equipped with a dexterous hand (i.e. a gripper with independent multi-joint fingers), it may also manipulate the object within its hand. However, these operations are very difficult to model and existing models still lack the required precision for constructing effective planners.

Movable objects allow new types of planning problems to be considered, where the important goals are not just to achieve given configurations of the robot. Arrangements of movable objects, such as assembling parts into a composite object or arranging pieces of furniture according to a given pattern, can now be the real goals.

6 Computational Complexity

Instances of the same type of motion planning problem may considerably differ in their "size", e.g. the dimension of configuration space and the

number of obstacles in the workspace. It is important to quantify the performance of the planning methods — typically, the time they require to solve a problem — and the inherent computational complexity of the planning problems as a function of these parameters. Analysis of the methods is useful to assess their practicality and to detect opportunities for increased efficiency. Analysis of the problems is useful to suggest new ways of formulating them when there is substantial evidence that the original formulation only admits too costly solutions.

In this section we review important theoretical results relative to the computational complexity of robot motion planning. We mainly consider lower bounds on the time complexity of problems. In the following chapters we will analyze the complexity of specific planning methods (upper bounds). Basic notions in computational complexity analysis are surveyed in Appendix B.

The first lower bound was established by Reif for planning free paths in a configuration space of arbitrary dimension ("generalized mover's problem") [Reif, 1979]:

Planning a free path for a robot made of an arbitrary number of polyhedral bodies connected together at some joint vertices, among a finite set of polyhedral obstacles, between any two given configurations, is a PSPACE-hard problem.

PSPACE-hardness has also been established for a variety of path planning problems with simpler or more specific robotic systems, for example:

- *Planar linkages.* A planar linkage consists of an arbitrary number of one-dimensional rigid links connected by revolute joints, each joint connecting two or more links. The locations of some joints are fixed. Although all the links move in the same plane, they are allowed to cross each other. In the absence of obstacles, deciding whether a planar linkage in some initial configuration can be moved so that a certain joint reaches a given point in the plane is PSPACE-hard [Hopcroft, Joseph, and Whitesides, 1984]. This result reflects the complexity of the connectivity of the configuration space of a planar linkage.

- *Multiple rectangles.* The robotic system consists of arbitrarily many rectangles in an empty rectangular workspace. Each rectangle can only move in translation with its sides parallel to the sides of the

workspace boundary. The problem is to plan the coordinated motion of the rectangles between two given configurations, so that they do not intersect. This problem, which is equivalent to planning a path in the configuration space of the multi-bodied robot consisting of all the rectangles, is both PSPACE-hard [Hopcroft, Schwartz and Sharir, 1984] and in PSPACE [Hopcroft and Wilfong, 1986b]. Hence, it is PSPACE-complete.

- *Planar arm.* Consider a planar arm consisting of arbitrarily many links serially connected by revolute joints such that all the links are constrained to move in the plane. Planning a free path for such an arm among a finite number of polygonal obstacles, between any two given configurations, is PSPACE-hard [Joseph and Plantiga, 1985].

The following result provides an upper bound on path planning complexity:

A free path in a configuration space of any fixed dimension m, when the free space is a set defined by n polynomial constraints of maximal degree d, can be computed by an algorithm whose time complexity is exponential in m and polynomial in both n ("geometrical complexity") and d ("algebraic complexity").

This bound was first established by Schwartz and Sharir [Schwartz and Sharir, 1983b], who gave an exact cell decomposition method based on the Collins decomposition algorithm. This method, which we will present in Chapter 5, is twice exponential in m. The most efficient algorithm to date is due to Canny [Canny, 1988] who proposed a roadmap method that is singly-exponential in m. Both methods reduce the planning problem to the problem of deciding the satisfiability of sentences in the first-order theory of the reals. As we will see in several chapters, the theory of the reals has been a powerful tool in establishing upper bounds for motion planning problems. Nonetheless, there exist no implementations of the decision algorithms that are fast enough to constitute practical solutions to these problems.

The above results strongly suggest that the complexity of path planning increases exponentially with the dimension of the configuration space. Although this statement is likely to be true for most robotic systems, there nevertheless exist simple settings for which it does not hold. For example, a polynomial time algorithm has been proposed to plan the

paths of a planar arm with arbitrarily many joints within a circle and without obstacles other than the circle [Hopcroft, Joseph and Whitesides, 1985] [Kantabutra and Kosaraju, 1986].

On the other hand, if the dimension of the configuration space is relatively small and/or we accept incomplete planning algorithms (i.e. algorithms that may fail to generate a path, even if one exists), specific algorithms of reasonable complexity can be designed. For example, if the dimension of configuration space is limited to two, it has been shown that, under reasonable assumptions, the basic motion planning problem can be solved in time $O(\lambda_s(n)\log^2 n)$, where n is the number of polynomial constraints defining the free space, s is a parameter depending in particular on the maximum degree of these constraints, and $\lambda_s(n)$ is an almost linear function of n (the length of (n, s) Davenport-Schinzel sequences) [Guibas, Sharir and Sifrony, 1988].

Interesting results have also been obtained with moving obstacles:

Planning the motion of a rigid object translating without rotation in three dimensions among arbitrarily many moving obstacles that may both translate and rotate is a PSPACE-hard problem if the velocity modulus of the object is bounded, and an NP-hard problem otherwise.

This result was established in [Reif and Sharir, 1985]. A simpler problem — planning the motion of a point in the plane, with bounded velocity modulus, among arbitrarily many convex polygonal obstacles moving at constant linear velocity without rotation — has also been shown to be NP-hard [Canny, 1988]. However, if the velocity of the robot is not constrained and if the motions of the obstacles can be described algebraically, the problem admits a polynomial time algorithm.

Concerning motion planning with uncertainty, the following result has been established by Canny and Reif [Canny and Reif, 1987]:

Planning compliant motions for a point in the presence of uncertainty, in a three-dimensional polyhedral configuration space with an arbitrarily large number of faces, is a nondeterministic exponential time hard (NEXPTIME-hard) problem.

This result means that the problem is expected to require at least exponential time on a nondeterministic machine, hence doubly-exponential time on a deterministic machine. Previously, Natarajan [Natarajan,

1986] had shown that this problem is PSPACE-hard. With a few assumptions allowing the reduction of the planning problem in an m-dimensional configuration space to a decision problem in the theory of the reals, an r-step plan (if any) can be generated by an algorithm of complexity $2^{2^{O(rnm)}}$, where n is the number of polynomial constraints defining the boundary of the C-obstacle region [Canny, 1989]. More tractable subclasses of motion planning in the plane in the presence of uncertainty have been identified.

7 Reduction of Complexity

The complexity results surveyed in the previous section give strong evidence that the time required to solve a motion planning problem increases quickly with the dimension of the configuration space, the number of polynomial constraints on the robot's motion, and the degree of these constraints. Therefore, it is important that these parameters be as small as possible.

To that purpose, one may approximate the real problem by a simplified, but still realistic problem, before submitting it to the motion planner. If the robot task is reasonably well understood and if it is acceptable to trade some generality against better time performance, such a simplification is almost always feasible. The simplified problems can be provided by the user, or automatically generated by the planner from the original input description.

The following subsections present techniques for approximating motion planning problems into simpler ones. Most of the discussion focuses on planning free paths for rigid objects and robot arms. However, some of the techniques can also be applied to other motion planning problems.

7.1 Projection in Configuration Space

One can reduce the dimension of the configuration space $\mathcal{C}$ by replacing the robot $\mathcal{A}$ by the surface or volume it sweeps out when it moves along r independent axes. This corresponds to projecting the m-dimensional configuration space $\mathcal{C}$ along r of its dimensions. Let $\mathcal{C}'$ be the projected space. Its dimension is $m - r$. The C-obstacles in $\mathcal{C}$ are projected into $\mathcal{C}'$, and path planning is conducted in $\mathcal{C}'_{free}$, the free space in $\mathcal{C}'$.

Examples: - Consider a two-dimensional object $\mathcal{A}$ that can both translate and rotate freely in the plane. $\mathcal{A}$ can be replaced by a disc $\mathcal{A}'$ containing $\mathcal{A}$ (ideally, this disc is bounded by the minimum spanning circle of $\mathcal{A}$, i.e. the smallest circle that encloses all the points of $\mathcal{A}$). This approximation reduces the configuration space dimension from 3 to 2. It is a realistic one only when $\mathcal{A}$ is not an elongated object.

- Consider a robot arm with six revolute joints, the axes of the last three joints intersecting at a single point. It is rather typical to replace the last three links, the end-effector and the payload by the volume they sweep out when the last three revolute joints span their angular ranges. This reduces the configuration space dimension from 6 to 3. This approximation is realistic only when the payload is not a long object. ∎

If a free path τ is found in $\mathcal{C}'$, it determines a family of free paths in $\mathcal{C}$. The configuration parameters not specified in τ can be selected arbitrarily. But, if no free path is found in $\mathcal{C}'$, this does not guarantee that there is no free path in $\mathcal{C}$.

If most of the motions of $\mathcal{A}$ occur in a rather uncluttered workspace and if the ratio of the area/volume of $\mathcal{A}$ by the area/volume of $\mathcal{A}'$ is close to 1, it is probably more efficient to plan a motion for $\mathcal{A}'$, rather than for $\mathcal{A}$. It is also possible to build a two-stage planner that plans the path of $\mathcal{A}$ only when it previously failed to plan the path of $\mathcal{A}'$.

7.2 Slicing in Configuration Space

Another way to reduce the dimension of configuration space is to consider a cross-section of dimension $m - r$ through $\mathcal{C}$ and to plan a free path in the space defined by this cross-section. This corresponds to forbidding the motion of $\mathcal{A}$ along r axes.

Examples: - Let $\mathcal{A}$ be a free-flying three-dimensional object. In order to plan a free path from the initial configuration $\mathbf{q}_{init} = (\mathcal{T}_{init}, \Theta_{init})$ to the goal configuration $\mathbf{q}_{goal} = (\mathcal{T}_{goal}, \Theta_{goal})$, one may consider the intermediate configuration $\mathbf{q}_{inter} = (\mathcal{T}_{goal}, \Theta_{init})$ at which $\mathcal{A}$ has the same orientation Θ_{init} as at the initial configuration and the same position $\mathcal{T}_{goal}$ as at the goal configuration. Then, the planner may attempt to generate a first path from $\mathbf{q}_{init}$ to $\mathbf{q}_{inter}$ with a fixed orientation of $\mathcal{A}$ and a second path from $\mathbf{q}_{inter}$ to $\mathbf{q}_{goal}$ with a fixed position of $\mathcal{A}$. Each of the two paths lies in a three-dimensional cross-section of the configuration

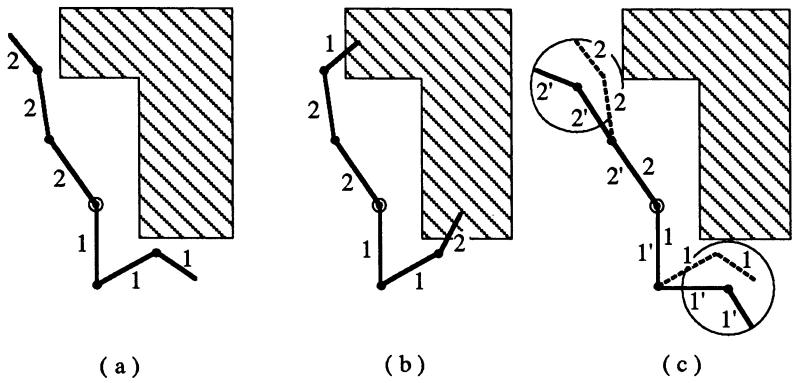

Figure 15. A planar three-joint arm has to move from the initial configuration marked **1** to the goal configuration marked **2**. Both configurations are shown in Figure a. The double-circled joint is fixed and there is a single obstacle. When the first two links are in the initial (resp. goal) configuration, and the third link in the goal (resp. initial) configuration, the arm collides with the obstacle as shown in Figure b. Therefore, the slicing technique alone does not solve this example. However, one can define two intermediate configurations (marked **1'** and **2'** in Figure c) which are close to the initial and goal configurations, respectively, such that the projection technique of Subsection 7.1 applies from **1'** to **2'**, and the slicing technique applies from **1** to **1'** and from **2'** to **2**.

space.

- Let us consider a slightly more complicated example, requiring the projection and slicing techniques to be combined. The robot is a planar arm with three revolute joints (see Figure 15). The initial and goal configurations of the arm, $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, are marked **1** and **2** in the figure. Neither the projection technique of Subsection 7.1 applied to the third joint, nor the slicing technique applied by rigidifying the third joint allows us to plan a free path. Indeed, the initial (resp. goal) orientation of the third link cannot be achieved without collision at the goal (resp. initial) orientation of the first two links. However, the projection technique applied to the third joint makes it possible to generate a path from a free configuration $\mathbf{q}'_{init}$ (marked **1'** in the figure) close to $\mathbf{q}_{init}$, to a free configuration $\mathbf{q}'_{goal}$ (marked **2'**) close to $\mathbf{q}_{goal}$. Let $\mathcal{C}'$ be the configuration space of the first two links. $\mathbf{q}'_{init}$ (resp. $\mathbf{q}'_{goal}$) can be selected as a configuration that (1) projects in the free subspace of $\mathcal{C}'$ at a configuration

close to the projection of $\mathbf{q}_{init}$ (resp. $\mathbf{q}_{goal}$), and (2) produces the same orientation of the third link with respect to the second link as $\mathbf{q}_{init}$ (resp. $\mathbf{q}_{goal}$). Then, it remains to plan a free path from $\mathbf{q}_{init}$ to $\mathbf{q}'_{init}$, and another one from $\mathbf{q}'_{goal}$ to $\mathbf{q}_{goal}$. This can be done by rigidifying the third joint and using the slicing technique. In the particular case of Figure 15, these two paths are rotations of the second joint. The orientation of the third link is changed along the path from $\mathbf{q}'_{init}$ to $\mathbf{q}'_{goal}$. ■

If planning in a cross-section of configuration space produces a free path, then this path is a free path in $\mathcal{C}$. But, if planning fails in a cross-section, this does not imply that there is no free path in $\mathcal{C}$.

7.3 Simplification of the Shape of Objects

The complexity of motion planning increases with the number of polynomial constraints defining the free space and the maximal degree of these constraints. One way to simplify these constraints is to approximate the shapes of both the robot and the obstacles by reducing the number of surfaces defining their boundary and/or the degree of the equations of these surfaces.

Simple and often efficient techniques consist of approximating the objects by rectangloids or balls. One can also approximate objects by more sophisticated polygons or polyhedra. This reduces the maximal degree of the constraints, but if one is tempted to increase the accuracy of the approximation, it ultimately results in an increase in the number of curves/surfaces, i.e. the number of constraints.

One may usefully distinguish between two kinds of approximation: *bounding* and *bounded* ones. A bounding approximation consists of replacing objects by new ones which completely contain the original ones. If a free path is generated with such an approximation, it is also a free path with the original objects (but the converse is not true). A bounded approximation consists of replacing objects by new objects completely contained in the original ones. If no free path is generated with such an approximation, it implies that there is no free path with the original objects (again, the converse is not true). Thus, bounded approximation can be used to detect that a particular problem has no solution.

One can generalize the above ideas and consider several bounding and bounded approximations of increasing precision. Then the planner it-

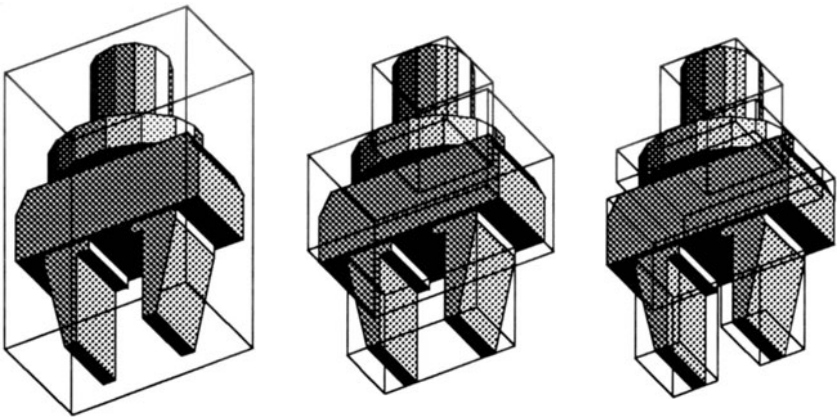

Figure 16. This figure shows three bounding approximations (from coarse to fine) of a robot gripper using rectangloids [Pasquier, 1989]. The motion planner may use such models in sequence until it finds a path.

eratively uses finer and finer approximations of the objects, until a free path is generated or there is evidence that no free path exists. Figure 16 illustrates a sequence of bounding approximations for a robot gripper.

7.4 Focusing Attention on a Subset of the Workspace

The techniques introduced in the three previous subsections are "simplification" heuristics. They exploit the fact that in most cases, motions take place in a relatively uncluttered space, so that there still remain solution paths after the various approximations have eliminated some. In many tasks, however, there are isolated localities that are quite cluttered and/or where contacts with obstacles have to be made (e.g. grasping, part mating). Simplification heuristics cannot be applied to these cluttered localities.

If cluttered localities can be identified in advance, one can break the original planning problem into subproblems. The subproblems involving cluttered localities only require a restricted subset of the workspace to be considered. Hence, the number of constraints on the motions is considerably reduced with respect to the original problem.

Example: Consider the problem of adding a part to an assembly with a robot arm. It can be broken down into five subproblems: (1) move the

arm to a configuration near the part, (2) grasp the part, (3) transfer the part to the current subassembly, (4) mate the part with the subassembly, and (5) move away to some rest configuration. In general, only subproblems (2) and (4) involve cluttered localities. In both subproblems, the motions usually have small amplitude, so that motion planning can be restricted to a small subset of the workspace. For instance, in (4), one can consider the part to be positioned (and perhaps also the gripper) as the robot, rather than the whole arm, and restrict the workspace of this object to a small volume around the assembly. If one wants to be sure that the generated plan is safe, the motion of the whole arm can then be checked for possible collision. In case a collision is possible, the planner would have to find another solution for (4). ∎

It can be further observed that in many cases cluttered localities are stereotyped situations (e.g. a mobile robot moving through a door or turning in a corridor). It may be worthwhile to attempt to recognize these situations and to apply specific (and more efficient) planning techniques to them. Along this line, Maddila and Yap explored the problems of moving a polygon around the corner in a corridor [Maddila and Yap, 1986] and through a door [Yap, 1987b]. Specific planning techniques applicable to stereotyped situations are called *local experts*.

8 Relation to Other Problems

An automatic motion planner can be used in a variety of ways. For instance, it can be included in an off-line robot programming system [Latombe et al., 1985] in order to simplify the task of describing robot trajectories. It can also be part of an interactive graphic process planning system (e.g. for planning the machining of a mechanical part, the assembly of an electromechanical device, or the construction of a power plant), and be used to check the geometrical feasibility of the planned operations (e.g. drilling a hole, mating two parts, erecting a pipe). More generally, it can be used to automatically generate graphic animated scenes. Nonetheless, beyond these "short-term" applications, the ultimate goal of automating motion planning is to create autonomous robots.

As mentioned earlier in this chapter, automatic path planning is a critical problem in the creation of autonomous robots, but it is not the only one. Other major problems include real-time motion control, sensing,

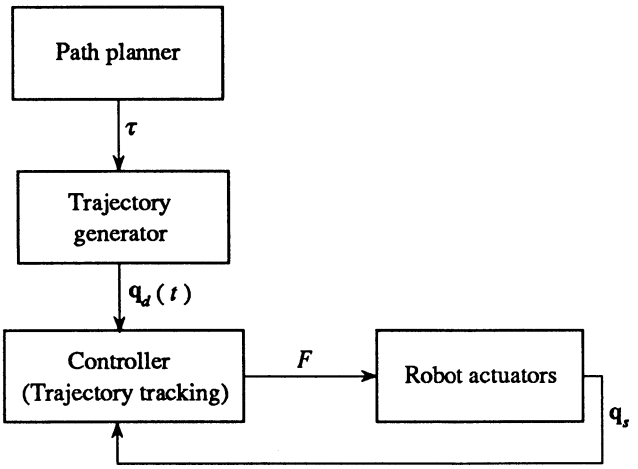

Figure 17. The path τ produced by the path planner is input into a trajectory generator that determines the time dependence of the configuration parameters. The output of the trajectory generator — the desired configurations q_d as a function of time — is fed into the controller. The controller computes the force F to be exerted by each actuator from the deviation of the current configuration q_s as measured by the sensors, relative to q_d.

and task-level planning. Because of the potentially complex interactions among them, the design of a truly autonomous robot will certainly require all these problems to be solved concurrently. But, today, our current understanding of the issues is insufficient to present a comprehensive picture of such a design. In the following subsections we therefore limit our presentation to a brief analysis of some possible interactions between motion planning and the three other problems mentioned above.

8.1 Interaction with Real-Time Motion Control

A motion plan specifies a motion to be executed by the robot. For example, a path specifies a continuous sequence of configurations that the robot should traverse toward the goal configuration. The basic task of the real-time controller is to make the robot execute the motion plan, e.g. follow the generated path.

Let us first assume that the motion plan is a free path τ. The task of

the controller is to transform τ into forces[5] to be exerted by the robot actuators. Typically, this transformation is broken down into two steps. The first step, called *trajectory generation*, consists of deciding on the velocity profile along the path. It can be done prior to motion execution. The second step, called *trajectory tracking*, consists of computing the forces to be exerted by the actuators at each time in order to perform the desired motion. The trajectory tracking step may use the dynamic equation of the robot — the equation expressing that the forces applied by the actuators equal the resultant of the various forces acting on the robot during motion, i.e. the gravitational, frictional, inertial, centrifugal and Coriolis forces — to compute the force that has to be delivered by each actuator. If the dynamic equation used by the controller was a perfect model, no feedback would be necessary. However, due to various disturbances, sensing is necessary to determine the deviation between the desired state and the actual state of the robot. While the motion is being executed, the controller computes the actuator forces which tend to eliminate this deviation at an update rate which typically ranges between 10 and 1000 Hz. Figure 17 shows the relationship between the path planner, the trajectory generator, the controller and the robot.

The simple architecture of Figure 17 is a classical one. Its advantage is to break the problem of deciding on a motion into clearly defined subproblems. But it may lead to rather inefficient motions. Even if the controller attempts to minimize execution time, the geometry of the paths may not allow the controller to fully exploit the capabilities of the actuators. A better approach is probably to take the dynamic equation of the robot, together with the actuators' saturation limits, into account during motion planning in order to generate paths allowing (quasi-)time-optimal motions to be executed. Several techniques have been recently proposed for implementing this approach [Donald and Xavier, 1989] [Barraquand, Langlois and Latombe, 1989a] [Shiller and Dubowsky, 1989] [Jacobs et al., 1989]. Such techniques may become particularly important when the robot operates in a dynamic workspace among mobile obstacles, when the task has to be accomplished within tight time bounds (e.g. because it interacts with an external process), and/or when the robot's productivity has to be optimized (e.g. in manufacturing).

[5]Here we use the word "force" in the generalized sense. It is either a force or a torque depending on the nature of the actuator.

Paths are the simplest sort of motion plans. We saw in Subsection 5.3 that in the presence of uncertainty, motion plans may include sensory-based motion commands, e.g. force-compliant motion commands, which are more tolerant to errors than classical position-controlled motion commands. The availability of such commands may make the task of the motion planner considerably simpler [Shekhar and Khatib, 1987]. The concurrent design of sensory-based motion primitives and motion planning methods using them would be a major step toward a better integration of motion planning and control. It has received little attention so far.

8.2 Interaction with Sensing

In the basic motion planning problem, we assumed that the robot has complete and exact knowledge of its environment. As mentioned in Subsection 5.3, this assumption is not always realistic. There are many real-life problems where the geometry of the workspace can only be partially known at planning time. Since planning requires the robot to have a model of its environment, the more incomplete this model, the more limited the role of planning.

If uncertainty in the prior knowledge is small, it is reasonable to anticipate all possible contingencies and to generate sensory-based motion plans that can deal with them (see Subsection 5.3). Sensing is used to guide the motions and monitor their execution. Planning makes sure that the motion commands will make enough information available at execution time so that the controller will be able to both monitor progress toward the goal and recognize goal achievement reliably.

On the other hand, if the robot has no prior knowledge about its environment, planning is useless. Then, the robot must rely heavily on sensing to acquire information while it is moving and it must react to the sensory data. For example, if the robot is equipped with proximity range sensors, a potential field method (see Subsection 4.3) can be applied on-line to guide the motion. The input goal configuration is used to create the attractive potential, while the data provided by the proximity sensors are used to create the repulsive potential. The negated gradient is treated as an external force, and the robot is controlled to comply to that force (i.e. to move along the flow of the negated gradient field) [Khatib, 1986]. There is no prior planning, and the robot can easily get

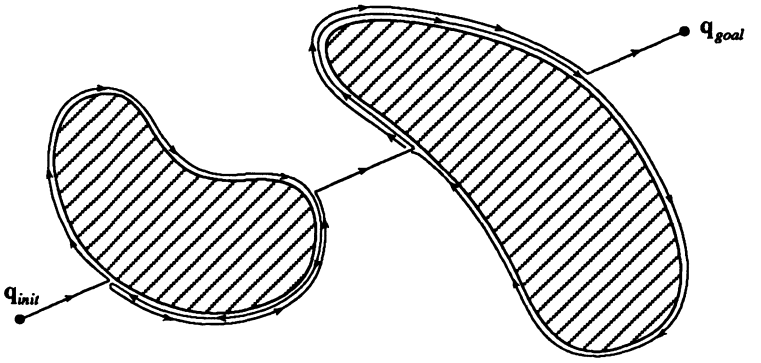

Figure 18. The robot has no prior knowledge of the obstacles. It follows the straight line segment joining q_{init} to q_{goal} until it hits an obstacle. Then it follows the C-obstacle's boundary, while keeping track of the intersections between this boundary and the line segment joining q_{init} and q_{goal}. When it comes back to the hitting point, the robot returns to the intersection point that is closest to q_{goal} by following the shortest path along the C-obstacle's boundary. From this point, it resumes moving along the straight line segment toward q_{goal}.

trapped into a local minimum of the potential field, e.g. a minimum created by an obstacle concavity. Escaping from such a local minimum requires some trial-and-error exploration by the robot.

Another reactive scheme has been proposed by Lumelsky when no prior knowledge is available [Lumelsky, 1987a]. The configuration space is assumed to be two-dimensional and the robot is equipped with a position sensor and a peripheral touch sensor that allows it to track the boundaries of the obstacles. Lumelsky described several specific algorithms which are variants of each other. One of them, illustrated in Figure 18, consists of having the robot move along a straight line toward the goal configuration q_{goal}. If the robot hits an obstacle, it shifts to moving along the boundary of the corresponding C-obstacle in a predetermined direction (e.g. clockwise) until it eventually comes back to the collision point. During this motion, it identifies the intersection point of the C-obstacle's boundary and the line segment joining the initial and goal configurations that is closest to the goal. After having made the closed-loop path around the C-obstacle it returns to this point by following the

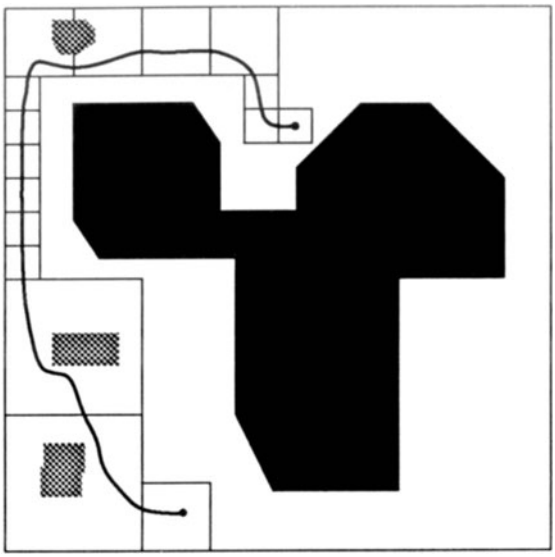

Figure 19. The region shown black is the C-obstacle region known at planning time. The regions shown grey depict unknown C-obstacles that are actually present. Knowledge available at planning time is used to compute a quadtree decomposition of the configuration space and extract a channel composed of adjacent cells lying outside the known C-obstacle region. At execution time, a sensory-based potential field is used to guide the robot through the channel away from the unexpected obstacles.

shortest path along the C-obstacle's boundary. From there it resumes moving along a straight line toward the goal. This simple algorithm is guaranteed to attain $\mathbf{q}_{goal}$ whenever every C-obstacle is bounded by a simple closed curve of finite length and any straight line segment intersects a finite number of C-obstacles. The length of the generated path is upper bounded by $D + 1.5\Sigma_i p_i$, where D is the distance between $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ and the p_i's are the perimeters of the C-obstacles intersecting the disc of radius D centered at $\mathbf{q}_{init}$. Some steps toward generalizing this kind of algorithm to configuration spaces of dimensions greater than two have been made in [Sun and Lumelsky, 1989] and [Cheung and Lumelsky, 1990].

The extreme case where the robot has no prior knowledge of its environment is rare. A more realistic situation is when the robot knows the

shape and location of the largest objects in its workspace, but has incomplete knowledge about smaller objects. An approach proposed in [Choi, Zhu and Latombe, 1989] to deal with such a situation is the following:

- first, a cell decomposition method is used to plan a channel among the known C-obstacles between the initial and goal configurations.
- Second, a potential field collision-avoidance method is used to guide the robot in the channel at execution time.

The potential field combines a dynamic attractive potential, generated by a target configuration that moves toward the goal configuration while the robot progresses in the channel, a repulsive potential, generated by the boundary of the channel (so that the robot's configuration does not escape the channel), and a repulsive potential, computed from the data provided by proximity range sensors. Figure 19 illustrates this approach in a two-dimensional configuration space. In this example, the channel was generated from a quadtree decomposition (the known C-obstacle region is shown black). The C-obstacles shown grey were not known at planning time and were detected during the motion. As long as the unexpected obstacles are small and sparsely distributed, the robot is unlikely to get trapped in a local minimum. However, there is no guarantee that this will never happen. When a local minimum is encountered, some replanning may be necessary using an updated model of the workspace.

There is a wide variety of possible software architecture interweaving motion planning and execution when prior knowledge is incomplete. The "best" architecture depends on both the nature and the extent of the lack of knowledge.

8.3 Interaction with Task-Level Planning

A motion planning goal is specified as a spatial arrangement of physical objects. Achieving such a goal contributes to the achievement of a more global task. For instance, the robot may be a troubleshooter whose task is to diagnose faults in pieces of electromechanical equipment and to repair them. Finding a fault typically requires the equipment to be inspected and operated, measuring devices to be connected, and probes to be inserted and removed. Each such operation can be specified by a spatial arrangement of objects, but deciding which operations must be

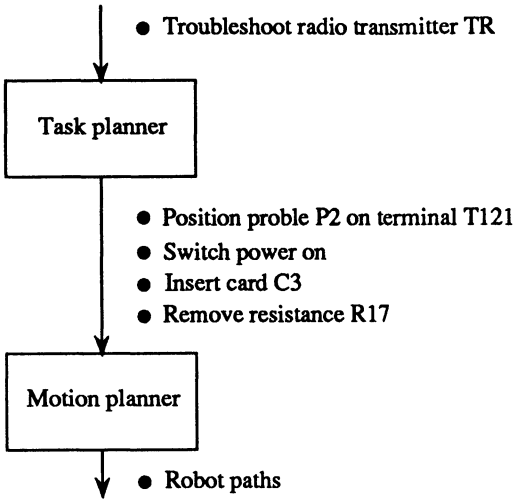

Figure 20. This figure illustrates the relationship between task planning and motion planning. The input to the task planner is a task description, e.g. "troubleshoot radio transmitter TR". Its output is a set of operations, e.g. "position proble P2 on terminal T121", "switch power on", "insert card C3", "remove resistance R17". Each operation can be specified as a goal spatial arrangement of objects (e.g. the goal position of the probe). The motion planner generates the motion plan for achieving this goal.

executed and their execution sequence — i.e. *task planning* — requires "higher-level" reasoning and planning capabilities than those possessed by a motion planner.

Although much research still remains to be done, Artificial Intelligence (AI) has produced a variety of methods for reasoning about tasks and for planning the high-level operations needed to accomplish them [Nilsson, 1980] [Genesereth and Nilsson, 1987] [Georgeff, 1987]. These methods are readily available to build planners whose outputs are sequences of operations, each specifying a new goal spatial relationship to the motion planner. Figure 20 illustrates the relationship between the task planner and the motion planner.

However, an AI task planner typically represents a task using logic-oriented constructs (e.g. predicate calculus language) which contain little information about the actual geometry of the objects and the spatial re-

lations among them. Therefore, a plan generated by the task planner may turn out to be infeasible when developed at the motion level. For instance, there may exist no path for a probe to the terminal at which it should be connected. In such a situation, the task planner would have to generate another plan. Important questions are: What kind of pertinent information should the motion planner feed back to the task planner? How can the task planner use this information to generate another plan?

In addition, the behavior of the equipment to be repaired might not be stationary, hence requiring the troubleshooting operations to be executed within strict time constraints. For instance, a probe may have to be moved from one position to another so that two measurements can be done within a short interval of time. However, the durations of the motions are typically not known with good precision until the motion plans have been generated.

This discussion suggests that integrating task planning and motion planning is a much more ambitious enterprise than Figure 20 suggests. Little has been done so far in this domain.

9 Literature Landmarks

We close this introductory chapter with a brief discussion of key publications that can be considered to be landmarks in the development of robot motion planning. The citations are given approximately in chronological order, so that this section may also be regarded as a short historical account of the development of the motion planning field over the past 20 years. Many other references will be given throughout the book.

Nilsson [Nilsson, 1969] described a mobile robot system (Shakey) with planning capabilities. Although most of the planning done by Shakey used a model of the workspace and the robot actions expressed in the predicate calculus language (Strips planner), Nilsson introduced the visibility graph method for planning Shakey's motions.

Ambler and Popplestone [Ambler and Popplestone, 1975] described a method for inferring the quantitative location of objects in terms of homogeneous coordinate matrices, from input spatial symbolic relations among features of objects. This method was extended later to a larger set of relations and implemented as part of a high-level robot program-

ming language (Rapt) [Popplestone, Ambler and Bellos, 1980].

Liebermann and Wesley [Liebermann and Wesley, 1977] and Lozano-Pérez [Lozano-Pérez, 1976] presented the first attempts to build integrated systems (Autopass and Lama, respectively) for automatically programming robot arms. The input to these systems was the description of a mechanical assembly, in the form of a set of geometrical models of the individual parts and goal assembly relations among the parts. The task of the systems was to automatically generate the robot programs for assembling the parts. Although these systems were never fully implemented, they have contributed in emphasizing the importance of geometrical reasoning in robot planning and in pointing out key motion planning problems in the context of mechanical assembly.

Taylor [Taylor, 1976] investigated the problem of planning motions in the presence of uncertainty. He proposed a method, known as "skeleton refinement", based on numerical uncertainty propagation techniques. A similar approach was also proposed in [Lozano-Pérez, 1976]. An improvement of the method, using symbolic propagation techniques rather than numerical ones, was later described by Brooks [Brooks, 1982].

Udupa [Udupa, 1977] introduced the idea of shrinking the robot to a point in an appropriate space (the term "configuration space" was not used yet). Lozano-Pérez and Wesley [Lozano-Pérez and Wesley, 1979] exploited this idea in a more systematic fashion, and proposed the first path planning algorithm for polygonal and polyhedral robots and obstacles without rotation. Their work is usually considered as the first contribution to "exact" motion planning. Later, Lozano-Pérez borrowed the notion of "configuration space" from Mechanics, and popularized it in motion planning. He also extended the ideas contained in [Lozano-Pérez and Wesley, 1979] and introduced the principle of the approximate cell decomposition approach [Lozano-Pérez, 1981 and 1983].

Reif [Reif, 1979] presented the first theoretical investigation of the inherent computational complexity of the path planning problem, showing that planning a free path in a configuration space of arbitrarily large dimension among fixed obstacles is PSPACE-hard. This work has contributed in attracting the interest of theoretical computer scientists toward motion planning. It has been followed by a flurry of papers on lower-bound complexity for a variety of motion planning problems (see

Section 6). Reif termed this area *Computational Robotics* [Reif, 1987].

Chatila [Chatila, 1982] investigated motion planning with incomplete knowledge for a mobile robot represented as a point in a two-dimensional workspace. He was the first to base his planner on an exact decomposition of the empty subset of the workspace into convex cells (an idea which was extended later to free space in configuration space). The planner was operating on-line, and the decomposition was periodically updated in order to take into account new information provided by the sensors.

Mason [Mason, 1982] stressed the importance of including pushing motions in a variety of manipulation tasks. Although his initial work was mainly aimed at modeling the mechanics of pushing under quasi-static assumptions, it has been followed by other publications by Mason and others directly related to motion planning.

Ó'Dúnlaing and Yap [Ó'Dúnlaing and Yap, 1982] introduced retraction as a new theoretical approach for path planning. Independently, and almost simultaneously, Brooks [Brooks, 1983a] presented a more empirical, "retraction-like" planning method, known as the freeway method.

In 1983 and 1984, Schwartz, Sharir, and Ariel-Sheffi published a series of five papers called the Piano Movers' Problem series [Schwartz and Sharir, 1983a, 1983b and 1983c] [Sharir and Ariel-Sheffi, 1983] [Schwartz and Sharir, 1983d]. The first paper of the series described the first exact method for planning free paths of a polygonal object allowed to both translate and rotate in a two-dimensional polygonal workspace. The second paper established the first upper bound on the time complexity of path planning in a semi-algebraic free space of any fixed dimension.

Laugier and Pertin [Laugier and Pertin, 1983] described an implemented method for planning the grasp position of a mechanical hand with two parallel fingers on an object bounded by planar, cylindrical, and spherical surfaces. In a first processing phase, the planner generates various grasps and assesses their local feasibility using a collection of empirical tests. In a second phase, it checks the global accessibility of each grasp by planning a path in the configuration space of the hand. These ideas were refined later by other authors.

Lozano-Pérez, Mason and Taylor [Lozano-Pérez, Mason and Taylor,

1984] described the preimage backchaining approach to motion planning with uncertainty in control and sensing. For the first time, this approach considered compliant motions in planning. In the original paper, the approach was essentially a theoretical framework, and was not implemented. Since then, the theory and the implementation of preimage backchaining have been investigated further.

Dufay and Latombe [Dufay and Latombe, 1984] described an implemented approach to motion planning with uncertainty based on inductive learning. The approach consists of executing the same task several times and combining the execution traces into a more general strategy. More recently, other researchers have investigated different learning techniques for motion planning and motion skill acquisition.

Gouzènes [Gouzènes, 1984a and 1984b] described the first implemented approximate cell decomposition method to plan the motion of robot arms with revolute joints. His method consists of approximating the free space as a collection of rectangloid cells obtained by dividing the range of possible angular positions of each link into small sub-intervals. The same type of method was later used and refined by several other researchers.

Khatib [Khatib, 1986] pioneered the potential field approach[6]. Although he implemented it as a real-time collision avoidance module in a robot controller, the approach is extendable to motion planning, which has been done since then. Koditschek [Koditschek, 1987] introduced the notion of a "navigation function", a local-minimum-free potential function. Using a variant of the potential field approach, Faverjon and Tournassoud [Faverjon and Tournassoud, 1987] implemented a practical system able to plan the motions of a manipulator arm with eight joints moving among vertical pipes. Barraquand and Latombe combined a potential field method with random techniques to escape from local minima and implemented a planner for generating free paths for robots with many degrees of freedom [Barraquand and Latombe, 1989a].

Laumond [Laumond, 1986 and 1987b] considered the problem of planning free paths for nonholonomic car-like mobile robots. He produced the interesting result that a free path for a free-flying robot in a two-

[6] Actually, Khatib introduced the principle of this approach in earlier publications (see [Khatib, 1980]). However, the paper cited here is the first comprehensive description of the approach.

dimensional workspace can always be transformed into a free path for a nonholonomic car-like robot having the same geometry and moving in the same workspace, by introducing simple backup maneuvers. Li and Canny [Li and Canny, 1989] were the first to apply tools developed in controllability theory for nonlinear systems to nonholonomic robots. These tools make it possible to generalize Laumond's results.

Lozano-Pérez et al. [Lozano-Pérez et al., 1987] described an implemented robot system, Handey, integrating a grasp planner and a path planner. This system is able to plan the motions of a manipulator robot for constructing simple assemblies made of polyhedral objects, and to execute the plans, assuming no uncertainty. Part of the initial knowledge about the workspace is obtained by using a vision system. In addition, the system is capable of planning the re-grasping of an object, if no grasp at the initial location of the object is compatible with a grasp at the goal location.

Exercises

1: Discuss in which respects motion planning differs from: (1) collision checking; (2) collision avoidance.

2: Design an algorithm for checking whether two polygons intersect.

3: Design an algorithm for checking whether two polyhedra intersect.

4: Consider the 3×3 matrix Θ defined in Subsection 3.1 for representing the orientation of a rigid object $\mathcal{A}$ in $\mathcal{W} = \mathbf{R}^3$. What equations relate the 9 parameters of the matrix? How many parameters are needed to represent a rotation? Comment on your results.

5: It is often said that representing the robot as a point in its configuration space transforms the problem of planning the motion of a dimensioned object into the "simpler" problem of planning the motion of a point. Do you agree or disagree with this assertion? Elaborate on your answer.

6: Let $\mathcal{C}$ be the configuration space of a compact rigid object $\mathcal{A}$ moving

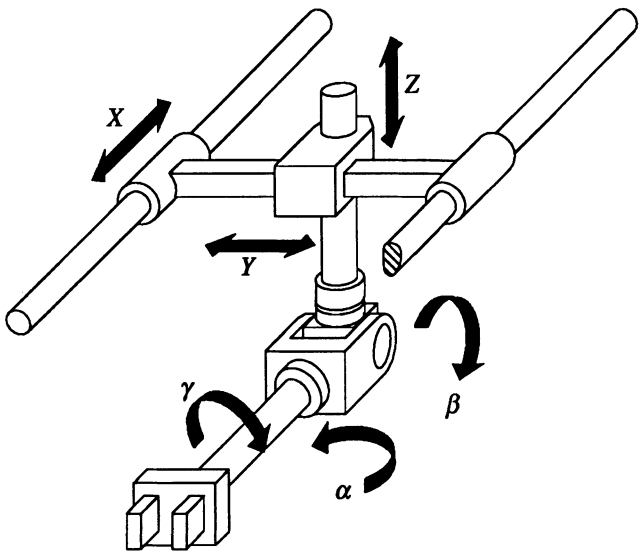

Figure 21. The first three joints of this robot are three perpendicular prismatic joints. The last three joints are revolute joints whose axes intersect at a single point.

freely in $\mathbf{R}^N$. Verify that the map $d : \mathcal{C} \times \mathcal{C} \to \mathbf{R}$ defined by:

$$\forall \mathbf{q}, \mathbf{q}' \in \mathcal{C} \ : \ d(\mathbf{q}, \mathbf{q}') = \max_{a \in \mathcal{A}} \|a(\mathbf{q}) - a(\mathbf{q}')\|,$$

where $\|x - x'\|$ denotes the Euclidean distance between two points x and x' in $\mathbf{R}^N$, determines a distance in $\mathcal{C}$.

7: Discuss the assumptions under which it is realistic to consider a wheeled mobile robot as a two-dimensional object moving freely in the plane.

8: Consider the simple six-joint gantry-type robot illustrated in Figure 21. The first three joints (denoted by X, Y and Z) are prismatic joints perpendicular to each other; the last three joints (denoted by α, β and γ) are revolute joints whose rotation axes intersect at a single point. Discuss the assumptions under which it is realistic to consider the end-effector of this robot as a free-flying object moving among obstacles in a three-dimensional workspace.

9: Consider the case where p free-flying objects $\mathcal{A}_1, ..., \mathcal{A}_p$ operate in the same workspace $\mathcal{W}$ among q fixed obstacles $\mathcal{B}_1, ..., \mathcal{B}_q$. Treat these moving objects as a multi-bodied robot $\mathcal{A} = \{\mathcal{A}_1, ..., \mathcal{A}_p\}$ whose configuration space is denoted by $\mathcal{C}$ (see Subsection 5.1). Specify the C-obstacles and the free space in $\mathcal{C}$ as we did for a single rigid object in Section 3.

10: Consider a point moving in $\mathbf{R}^2$. Assume that its motions are imperfectly controlled, but that its actual position is always within a distance less than a known constant ε from the position where it is desired to be. How must the obstacles be grown so that path planning is guaranteed to generate a path whose execution will be collision-free? Illustrate with an example the need for more sophisticated motion planning methods capable of generating sensory-based motion plans, rather than just paths.

11: Consider a part mating operation. What are the sources of errors that may prevent a position-controlled motion to execute the operation successfully? Is it reasonable to bound these errors by uncertainty intervals and to plan a robust motion plan as suggested in Subsection 5.3?

12: Propose and discuss one or two extensions of the basic motion planning problem not presented in Section 5.

13: Define (in a sentence) the words "kinematics" and "dynamics". [Hint: Look into a good dictionary.]

14: Identify and discuss briefly some typical motion planning problems for: (1) a mobile robot carrying objects in an office environment; (2) a space platform robot actuated by thrusters and equipped with two arms for assembling an orbital platform.

15: What does it mean for an algorithm to have time complexity $O(1)$? $n^{O(1)}$? $O(2^n)$? $2^{O(n)}$?

16: Sketch a few techniques for approximating objects by simple geometrical models and possibly reducing the complexity of motion planning.

17: Compare the projection and slicing techniques for reducing the di-

mension of configuration space (see Subsections 7.1 and 7.2). Are they *both* useful? Why?

18: Propose a software architecture for interweaving motion planning and motion execution when prior knowledge is incomplete.

19: Compare motion planning to the type of planning performed by an Artificial Intelligence planner such as Strips [Nilsson, 1980]. Describe, in general terms, a system integrating the two types of planning.