Chapter 7

PATH PLANNING AND COLLISION AVOIDANCE

In previous chapters, we have studied the geometry of robot arms, developing solutions for both the forward and inverse kinematics problems. The solutions to these problems depend only on the intrinsic geometry of the robot, and they do not reflect any constraints imposed by the workspace in which the robot operates. In particular, they do not take into account the possiblity of collision between the robot and objects in the workspace. In this chapter, we address the problem of planning collision free paths for the robot. We will assume that the initial and final configurations of the robot are specified, and that the problem is to find a collision free path for the robot that connects them.

The description of this problem is deceptively simple, yet the path planning problem is among the most difficult problems in computer science. The computational complexity of the best known complete[1] path planning algorithm grows exponentially with the number of internal degrees of freedom of the robot. For this reason, for robot systems with more than a few degrees of freedom, complete algorithms are not used in practice.

In this chapter, we treat the path planning problem as a search problem. The algorithms we describe are not guaranteed to find a solution to all problems, but they are quite effective in a wide range of practical applications. Furthermore, these algorithms are fairly easy to implement, and require only moderate computation time for most problems.

We begin in Section 7.1 by introducing the notion of *configuration space*, and describing how obstacles in the workspace can be mapped into the

[1]An algorithm is said to be complete if it finds a solution whenever one exists.

configuration space. We then introduce path planning methods that use artificial potential fields in Sections 7.2 and 7.3. The corresponding algorithms use gradient descent search to find a collision-free path to the goal, and, as with all gradient descent methods, these algorithms can become trapped in local minima in the potential field. Therefore, in Section 7.4 we describe how random motions can be used to escape local minima. In Section 7.5 we describe another randomized method known as the Probabilistic Roadmap (PRM) method. Finally, since each of these methods generates a sequence of configurations, we describe in Chapter 8 how polynomial splines can be used to generate smooth trajectories from a sequence of configurations.

As in previous chapters, we will restrict our attention in this chapter to the geometric aspects of the problem. In future chapters we will consider the problem of generating motor torques to execute such a trajectory.

7.1 The Configuration Space

In Chapter 3, we learned that the forward kinematic map can be used to determine the position and orientation of the end effector frame given the vector of joint variables. Furthermore, the A_i matrices can be used to infer the position and orientation of the reference frame for any link of the robot. Since each link of the robot is assumed to be a rigid body, the A_i matrices can therefore be used to infer the position of any point on the robot, given the values of the joint variables. In the path planning literature, a complete specification of the location of every point on the robot is referred to as a *configuration*, and the set of all possible configurations is referred to as the *configuration space*. For our purposes, the vector of joint variables, $\boldsymbol{q}$, provides a convenient representation of a configuration. We will denote the configuration space by $\mathcal{Q}$.

For a one link revolute arm, the configuration space is merely the set of orientations of the link, and thus $\mathcal{Q} = S^1$, where S^1 represents the unit circle. We could also say $\mathcal{Q} = SO(2)$. In fact, the choice of S^1 or $SO(2)$ is not particularly important, since these two are equivalent representations. In either case, we can parameterize $\mathcal{Q}$ by a single parameter, the joint angle θ_1. For the two-link planar arm, we have $\mathcal{Q} = S^1 \times S^1 = T^2$, in which T^2 represents the torus, and we can represent a configuration by $\boldsymbol{q} = (\theta_1, \theta_2)$. For a Cartesian arm, we have $\mathcal{Q} = \Re^3$, and we can represent a configuration by $\boldsymbol{q} = (d_1, d_2, d_3) = (x, y, z)$.

Although we have chosen to represent a configuration by a vector of joint variables, the notion of a configuration is more general than this. For example, as we saw in Chapter 2, for any rigid two-dimensional object, we can specify the location of every point on the object by rigidly attaching a coordinate frame to the object, and then specifying the position and orientation of this frame. Thus, for a rigid object moving in the plane we can represent a configuration by the triple $\boldsymbol{q} = (x, y, \theta)$, and the configuration space can be represented by $\mathcal{Q} = \Re^2 \times SO(2)$. Again, this is merely one possible representation of the configuration space, but it is a convenient one given the representations of position and orientation that we have learned in preceeding chapters.

A collision occurs when the robot contacts an obstacle in the workspace. To describe collisions, we introduce some additional notation. We will denote the robot by $\mathcal{A}$, and by $\mathcal{A}(\boldsymbol{q})$ the subset of the workspace that is occupied by the robot at configuration $\boldsymbol{q}$. We denote by $\mathcal{O}_i$ the obstacles in the workspace, and by $\mathcal{W}$ the workspace (i.e., the Cartesian space in which the robot moves). To plan a collision free path, we must ensure that the robot never reaches a configuration $\boldsymbol{q}$ that causes it to make contact with an obstacle in the workspace. The set of configurations for which the robot collides with an obstacle is referred to as the configuration space obstacle, and it is defined by

$$\mathcal{QO} = \{\boldsymbol{q} \in \mathcal{Q} \mid \mathcal{A}(\boldsymbol{q}) \cap \mathcal{O} \neq \emptyset\}.$$

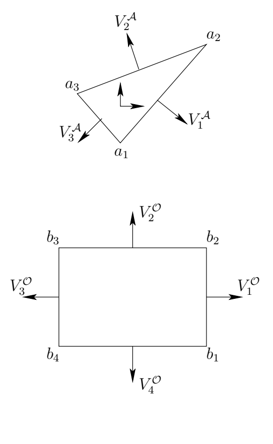
Here, we define $\mathcal{O} = \cup \mathcal{O}_i$. The set of collision-free configurations, referred to as the free configuration space, is then simply

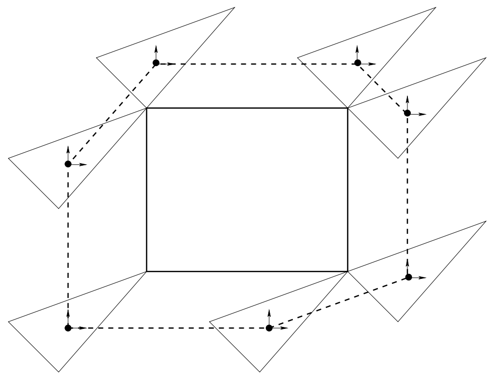
$$\mathcal{Q}_{\text{free}} = \mathcal{Q} \setminus \mathcal{QO}.$$

Example 7.1 *A Rigid Body that Translates in the Plane.*

Consider a simple gantry robot with two prismatic joints and forward kinematics given by $x = d_1, y = d_2$. For this case, the robot's configuration space is $\mathcal{Q} = \Re^2$, so it is particularly easy to visualize both the configuration space and the configuration space obstacle region. If there is only one obstacle in the workspace and both the robot end-effector and the obstacle are convex polygons, it is a simple matter to compute the configuration space obstacle region, $\mathcal{QO}$ (we assume here that the arm itself is positioned above the workspace, so that the only possible collisions are between the end-effector and obstacles the obstacle).

Let $V_i^{\mathcal{A}}$ denote the vector that is normal to the i^{th} edge of the robot and $V_i^{\mathcal{O}}$ denote the vector that is normal to the i^{th} edge of the obstacle. Define a_i to be the vector from the origin of the robot's coordinate frame to the





(a) (b)

Figure 7.1: (a) a rigid body, $\mathcal{A}$, in a workspace containing a single rectangular obstacle, $\mathcal{O}$, (b) illustration of the algorithm to construct $\mathcal{QO}$, with the boundary of $\mathcal{QO}$ shown as the dashed line.

i^{th} vertex of the robot and b_j to be the vector from the origin of the world coordinate frame to the j^{th} vertex of the obstacle. An example is shown in Figure 7.1(a). The vertices of $\mathcal{QO}$ can be determined as follows.

- *For each pair $V_j^{\mathcal{O}}$ and $V_{j-1}^{\mathcal{O}}$, if $V_i^{\mathcal{A}}$ points between $-V_j^{\mathcal{O}}$ and $-V_{j-1}^{\mathcal{O}}$ then add to $\mathcal{QO}$ the vertices $b_j - a_i$ and $b_j - a_{i+1}$.*
- *For each pair $V_i^{\mathcal{A}}$ and $V_{i-1}^{\mathcal{A}}$, if $V_j^{\mathcal{O}}$ points between $-V_i^{\mathcal{A}}$ and $-V_{i-1}^{\mathcal{A}}$ then add to $\mathcal{QO}$ the vertices $b_j - a_i$ and $b_{j+1} - a_i$.*

This is illustrated in Figure 7.1(b). Note that this algorithm essentially places the robot at all positions where vertex-vertex contact between robot and obstacle are possible. The origin of the robot's local coordinate frame at each such configuration defines a vertex of $\mathcal{QO}$. The polygon defined by these vertices is $\mathcal{QO}$.

If there are multiple convex obstacles $\mathcal{O}_i$, then the configuration space obstacle region is merely the union of the obstacle regions $\mathcal{QO}_i$, for the individual obstacles. For a nonconvex obstacle, the configuration space obstacle region can be computed by first decomposing the nonconvex obstacle into convex pieces, $\mathcal{O}_i$, computing the C-space obstacle region, $\mathcal{QO}_i$ for each piece, and finally, computing the union of the $\mathcal{QO}_i$.

◇

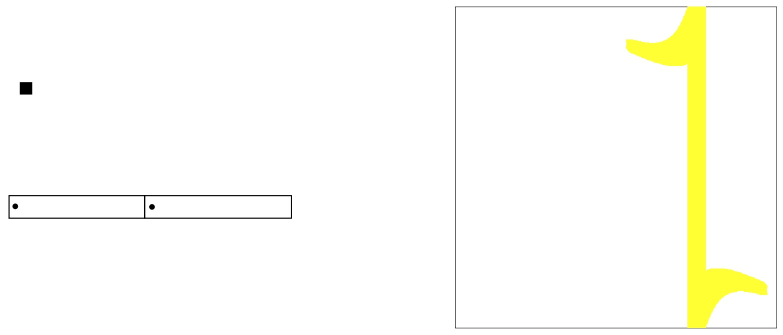

(a) (b)

Figure 7.2: (a) A two-link planar arm and a single polygonal obstacle. (b) The corresponding configuration space obstacle region.

Example 7.2 *A Two Link Planar Arm.*

For robots with revolute joints, computation of $\mathcal{QO}$ is more difficult. Consider a two-link planar arm in a workspace containing a single obstacle as shown in Figure 7.2(a). The configuration space obstacle region is illustrated in 7.2(b). The horizontal axis in 7.2(b) corresponds to θ_1, and the vertical axis to θ_2. For values of θ_1 very near $\pi/2$, the first link of the arm collides with the obstacle. Further, when the first link is near the obstacle (θ_1 near $\pi/2$), for some values of θ_2 the second link of the arm collides with the obstacle. The region $\mathcal{QO}$ shown in 7.2(b) was computed using a discrete grid on the configuration space. For each cell in the grid, a collision test was performed, and the cell was shaded when a collision occured. Thus, this is only an approximate representation of $\mathcal{QO}$.

$\diamond$

Computing $\mathcal{QO}$ for the two-dimensional case of $\mathcal{Q} = \Re^2$ and polygonal obstacles is straightforward, but, as can be seen from the two-link planar arm example, computing $\mathcal{QO}$ becomes difficult for even moderately complex configuration spaces. In the general case (e.g., articulated arms or rigid bodies that can both translate and rotate), the problem of computing a representation of the configuration space obstacle region is intractable. One of the reasons for this complexity is that the size of the representation of the configuration space tends to grow exponentially with the number of degrees of freedom. This is easy to understand intuitively by considering the number of n-dimensional unit cubes needed to fill a space of size k. For the one dimensional case, k unit intervals will cover the space. For the two-dimensional case, k^2 squares are required. For the three-dimensional case, k^3 cubes are required, and so on. Therefore, in this chapter we will develop methods that avoid the construction of an explicit representation of $\mathcal{Q}_{\text{free}}$.

The path planning problem is to find a path from an initial configuration $\boldsymbol{q}_{\text{init}}$ to a final configuration $\boldsymbol{q}_{\text{final}}$, such that the robot does not collide with any obstacle as it traverses the path. More formally, A collision-free path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$ is a continuous map, $\tau : [0,1] \rightarrow \mathcal{Q}_{\text{free}}$, with $\tau(0) = \boldsymbol{q}_{\text{init}}$ and $\tau(1) = \boldsymbol{q}_{\text{final}}$. We will develop path planning methods that compute a sequence of discrete configurations (set points) in the configuration space. In Chapter 8 we will show how smooth trajectories can be generated from such a sequence.

7.2 Path Planning Using Configuration Space Potential Fields

As mentioned above, it is not feasible to build an explicit representation of $\mathcal{Q}_{\text{free}}$. An alternative is to develop a search algorithm that incrementally explores $\mathcal{Q}_{\text{free}}$ while searching for a path. Such a search algorithm requires a strategy for exploring $\mathcal{Q}_{\text{free}}$, and one of the most popular is to use an *artificial potential field* to guide the search. In this section, we will introduce artificial potential field methods. Here we describe how the potential field can be constructed directly on the configuration space of the robot. However, as will become clear, computing the gradient of such a field is not feasible in general, so in Section 7.3 we will develop an alternative, in which the potential field is first constructed on the workspace, and then its effects are mapped to the configuration space.

The basic idea behind the potential field approaches is as follows. The robot is treated as a point particle in the configuration space, under the influence of an artificial potential field U. The field U is constructed so that the robot is attracted to the final configuration, $\boldsymbol{q}_{\text{final}}$, while being repelled from the boundaries of $\mathcal{QO}$. If U is constructed appropriately, there will be a single global minimum of U at $\boldsymbol{q}_{\text{final}}$, and no local minima. Unfortunately, as we will discuss below, it is often difficult to construct such a field.

In general, the field U is an additive field consisting of one component that attracts the robot to $\boldsymbol{q}_{\text{final}}$ and a second component that repels the robot from the boundary of $\mathcal{QO}$,

$$U(\boldsymbol{q}) = U_{\text{att}}(\boldsymbol{q}) + U_{\text{rep}}(\boldsymbol{q}). \tag{7.1}$$

Given this formulation, path planning can be treated as an optimization problem, i.e., find the global minimum in U, starting from initial configuration $\boldsymbol{q}_{\text{init}}$. One of the easiest algorithms to solve this problem is gradient descent. In this case, the negative gradient of U can be considered as a force acting on the robot (in configuration space),

$$F(\boldsymbol{q}) = -\nabla U(\boldsymbol{q}) = -\nabla U_{\text{att}}(\boldsymbol{q}) - \nabla U_{\text{rep}}(\boldsymbol{q}). \tag{7.2}$$

In the remainder of this section, we will describe typical choices for the attractive and repulsive potential fields, and a gradient descent algorithm that can be used to plan paths in this field.

7.2.1 The Attractive Field

There are several criteria that the potential field U_{att} should satisfy. First, U_{att} should be monotonically increasing with distance from $\boldsymbol{q}_{\text{final}}$. The simplest choice for such a field is a field that grows linearly with the distance from $\boldsymbol{q}_{\text{final}}$, a so-called conic well potential. However, the gradient of such a field has unit magnitude everywhere but the origin, where it is zero. This can lead to stability problems, since there is a discontinuity in the attractive force at the origin. We prefer a field that is continuously differentiable, such that the attractive force decreases as the robot approaches $\boldsymbol{q}_{\text{final}}$. The simplest such field is a field that grows quadratically with the distance to $\boldsymbol{q}_{\text{final}}$. Let $\rho_f(\boldsymbol{q})$ be the Euclidean distance between $\boldsymbol{q}$ and $\boldsymbol{q}_{\text{final}}$, i.e., $\rho_f(\boldsymbol{q}) = ||\boldsymbol{q} - \boldsymbol{q}_{\text{final}}||$. Then we can define the quadratic field by

$$U_{\text{att}}(\boldsymbol{q}) = \frac{1}{2}\zeta \rho_f^2(\boldsymbol{q}), \tag{7.3}$$

in which ζ is a parameter used to scale the effects of the attractive potential. This field is sometimes referred to as a parabolic well. For $\boldsymbol{q} = (q^1, \cdots q^n)^T$, the gradient of U_{att} is given by

$$\begin{aligned}
\nabla U_{\text{att}}(\boldsymbol{q}) &= \nabla \frac{1}{2}\zeta\rho_f^2(\boldsymbol{q}) && (7.4)\\
&= \nabla \frac{1}{2}\zeta||\boldsymbol{q}-\boldsymbol{q}_{\text{final}}||^2 && (7.5)\\
&= \frac{1}{2}\zeta\nabla\sum(q^i - q^i_{\text{final}})^2 && (7.6)\\
&= \zeta(q^1 - q^1_{\text{final}}, \cdots, q^n - q^n_{\text{final}})^T && (7.7)\\
&= \zeta(\boldsymbol{q}-\boldsymbol{q}_{\text{final}}). && (7.8)
\end{aligned}$$

Here, (7.7) follows since

$$\frac{\partial}{\partial q^j}\sum_i (q^i - q^i_{\text{final}})^2 = 2(q^j - q^j_{\text{final}}).$$

So, for the parabolic well, the attractve force, $F_{\text{att}}(q) = -\nabla U_{\text{att}}(\boldsymbol{q})$ is a vector directed toward $\boldsymbol{q}_{\text{final}}$ with magnitude linearly related to the distance from $\boldsymbol{q}$ to $\boldsymbol{q}_{\text{final}}$.

Note that while $F_{\text{att}}(\boldsymbol{q})$ converges linearly to zero as $\boldsymbol{q}$ approaches $\boldsymbol{q}_{\text{final}}$ (which is a desirable property), it grows without bound as $\boldsymbol{q}$ moves away from $\boldsymbol{q}_{\text{final}}$. If $\boldsymbol{q}_{\text{init}}$ is very far from $\boldsymbol{q}_{\text{final}}$, this may produce an attractive force that is too large. For this reason, we may choose to combine the quadratic and conic potentials so that the conic potential attracts the robot when it is very distant from $\boldsymbol{q}_{\text{final}}$ and the quadratic potential attracts the robot when it is near $\boldsymbol{q}_{\text{final}}$. Of course it is necessary that the gradient be defined at the boundary between the conic and quadratic portions. Such a field can be defined by

$$U_{\text{att}}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\zeta\rho_f^2(\boldsymbol{q}) & : \ \rho_f(\boldsymbol{q}) \le d \\ \\ d\zeta\rho_f(\boldsymbol{q}) - \frac{1}{2}\zeta d^2 & : \ \rho_f(\boldsymbol{q}) > d \end{cases} . \qquad (7.9)$$

and in this case we have

$$F_{\text{att}}(q) = -\nabla U_{\text{att}}(\boldsymbol{q}) = \begin{cases} -\zeta(\boldsymbol{q}-\boldsymbol{q}_{\text{final}}) & : \ \rho_f(\boldsymbol{q}) \le d \\ \\ -\dfrac{d\zeta(\boldsymbol{q}-\boldsymbol{q}_{\text{final}})}{\rho_f(\boldsymbol{q})} & : \ \rho_f(\boldsymbol{q}) > d \end{cases} . \qquad (7.10)$$

The gradient is well defined at the boundary of the two fields since at the boundary $d = \rho_f(\boldsymbol{q})$, and the gradient of the quadratic potential is equal to the gradient of the conic potential, $F_{\text{att}}(q) = -\zeta(\boldsymbol{q}-\boldsymbol{q}_{\text{final}})$.

7.2.2 The Repulsive field

There are several criteria that the repulsive field should satisfy. It should repel the robot from obstacles, never allowing the robot to collide with an obstacle, and, when the robot is far away from an obstacle, that obstacle should exert little or no influence on the motion of the robot. One way to achieve this is to define a potential that goes to infinity at obstacle boundaries, and drops to zero at a certain distance from the obstacle. If we define ρ_0 to be the distance of influence of an obstace (i.e., an obstacle will not repel the robot if the distance from the robot to the obstacle is greater that ρ_0), one potential that meets these criteria is given by

$$U_{\text{rep}}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)^2 & : \quad \rho(\boldsymbol{q}) \le \rho_0 \\ 0 & : \quad \rho(\boldsymbol{q}) > \rho_0 \end{cases} \tag{7.11}$$

in which $\rho(\boldsymbol{q})$ is the shortest distance from $\boldsymbol{q}$ to a configuration space obstacle boundary, and η is a scalar gain coefficient that determines the influence of the repulsive field. If $\mathcal{QO}$ consists of a single convex region, the corresponding repulsive force is given by the negative gradient of the repulsive field,

$$F_{\text{rep}}(\boldsymbol{q}) = \begin{cases} \eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(\boldsymbol{q})}\nabla\rho(\boldsymbol{q}) & : \quad \rho(\boldsymbol{q}) \le \rho_0 \\ 0 & : \quad \rho(\boldsymbol{q}) > \rho_0 \end{cases} . \tag{7.12}$$

When $\mathcal{QO}$ is convex, the gradient of the distance to the nearest obstacle is given by

$$\nabla\rho(\boldsymbol{q}) = \frac{\boldsymbol{q} - \boldsymbol{b}}{||\boldsymbol{q} - \boldsymbol{b}||}, \tag{7.13}$$

in which $\boldsymbol{b}$ is the point in the boundary of $\mathcal{QO}$ that is nearest to $\boldsymbol{q}$. The derivation of (7.12) and (7.13) are left as exercises 2.

If $\mathcal{QO}$ is not convex, then ρ won't necessarily be differentiable everywhere, which implies discontinuity in the force vector. Figure 7.3 illustrates such a case. Here $\mathcal{QO}$ contains two rectangular obstacles. For all configurations to the left of the dashed line, the force vector points to the right, while for all configurations to the right of the dashed line the force vector points to the left. Thus, when the configuration of the robot crosses the dashed

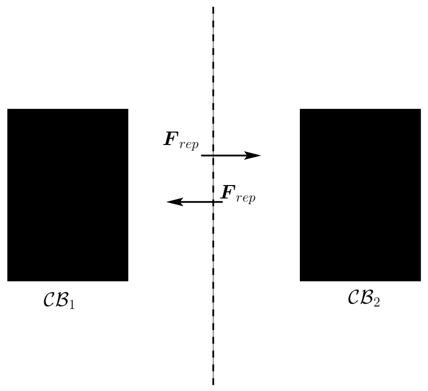


Figure 7.3: Situation in which the gradient of the repuslive potential of (7.11) is not continuous.

line, a discontinuity in force occurs. There are various ways to deal with this problem. The simplest of these is merely to ensure that the regions of influence of distinct obstacles do not overlap.

7.2.3 Gradient Descent Planning

Gradient descent is a well known approach for solving optimization problems. The idea is simple. Starting at the initial configuration, take a small step in the direction of the negative gradient (i.e., in the direction decreases the potential as quickly as possible). This gives a new configuration, and the process is repeated until the final configuration is reached. More formally, we can define a gradient descent algorithm as follows.

```
1.  $\boldsymbol{q}^0 \leftarrow \boldsymbol{q}_{\text{init}},\ i \leftarrow 0$
2.  **IF** $\boldsymbol{q}^i \neq \boldsymbol{q}_{\text{final}}$
        $\boldsymbol{q}^{i+1} \leftarrow \boldsymbol{q}^i + \alpha^i \frac{F(\boldsymbol{q}^i)}{||F(\boldsymbol{q}^i)||}$
        $i \leftarrow i + 1$
    **ELSE** return $< \boldsymbol{q}^0, \boldsymbol{q}^1 \cdots \boldsymbol{q}^i >$
3.  **GO TO** 2
```

In this algorithm, the notation $\boldsymbol{q}^i$ is used to denote the value of $\boldsymbol{q}$ at the i^{th} iteration (not the i^{th} componenent of the vector $\boldsymbol{q}$) and the final path consists of the sequence of iterates $< \boldsymbol{q}^0, \boldsymbol{q}^1 \cdots \boldsymbol{q}^i >$. The value of the scalar α^i determines the step size at the i^{th} iteration; it is multiplied by the unit vector in the direction of the resultant force. It is important that α^i be small

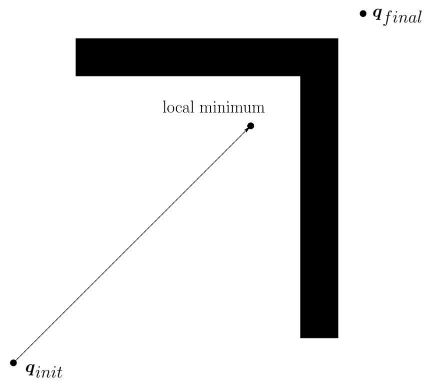


Figure 7.4: This figure illustrates the progress of a gradient descent algorithm from $\boldsymbol{q}_{\text{init}}$ to a local minimum in the field U.

enough that the robot is not allowed to “jump into” obstacles, while being large enough that the algorithm doesn't require excessive computation time. In motion planning problems, the choice for α^i is often made on an ad hoc or empirical basis, perhaps based on the distance to the nearest obstacle or to the goal. A number of systematic methods for choosing α^i can be found in the optimization literature [?]. The constants ζ and η used to define U_{att} and U_{rep} essentially arbitrate between attractive and repulsive forces. Finally, it is unlikely that we will ever exactly satisfy the condition $\boldsymbol{q}^i = \boldsymbol{q}_{\text{final}}$. For this reason, this condition is often replaced with the more forgiving condition $||\boldsymbol{q}^i - \boldsymbol{q}_{\text{final}}|| < \epsilon$, in which ϵ is chosen to be sufficiently small, based on the task requirements.

The problem that plagues all gradient descent algorithms is the possible existence of local minima in the potential field. For appropriate choice of α^i, it can be shown that the gradient descent algorithm is guaranteed to converge to a minimum in the field, but there is no guarantee that this minimum will be the global minimum. In our case, this implies that there is no guarantee that this method will find a path to $\boldsymbol{q}_{\text{final}}$. An example of this situation is shown in Figure 7.4. We will discuss ways to deal this below in Section 7.4.

One of the main difficulties with this planning approach lies in the evaluation of ρ and $\nabla\rho$. In the general case, in which both rotational and translational degrees of freedom are allowed, this becomes even more difficult. We address this general case in the next section.

7.3 Planning Using Workspace Potential Fields

As described above, in the general case, it is extremely difficult to compute an explicit representation of $\mathcal{QO}$, and thus it can be extremely difficult to compute ρ and $\nabla\rho$. In fact, in general for a curved surface there does not exist a closed form expression for the distance from a point to the surface. Thus, even if we had access to an explicit representation of $\mathcal{QO}$, it would still be difficult to compute ρ and $\nabla\rho$ in (7.12). In order to deal with these difficulties, in this section we will modify the potential field approach of Section 7.2 so that the potential function is defined on the workspace, $\mathcal{W}$, instead of the configuration space, $\mathcal{Q}$. Since $\mathcal{W}$ is a subset of a low dimensional space (either $\Re^2$ or $\Re^3$), it will be much easier to implement and evaluate potential functions over $\mathcal{W}$ than over $\mathcal{Q}$.

We begin by describing a method to define an appropriate potential field on the workspace. This field should have the properties that the potential is at a minimum when the robot is in its goal configuration, and the potential should grow without bound as the robot approaches an obstacle. As above, we will define a global potential field that is the sum of attractive and repulsive fields. Once we have constructed the workspace potential, we will develop the tools to map its gradient to changes in the joint variable values (i.e., we will map workspace forces to changes in configuration). Finally, we will present a gradient descent algorithm similar to the one presented above, but which can be applied to robots with more complicated kinematics.

7.3.1 Defining Workspace Potential Fields

As before, our goal in defining potential functions is to construct a field that repels the robot from obstacles, with a global minimum that corresponds to $\boldsymbol{q}_{\text{final}}$. In the configuration space, this task was conceptually simple because the robot was represented by a single point, which we treated as a point mass under the influence of a potential field. In the workspace, things are not so simple; the robot is an articulated arm with finite volume. Evaluating the effect of a potential field over the arm would involve computing an integral over the volume of the arm, and this can be quite complex (both mathematically and computationally). An alternative approach is to select a subset of points on the robot, called control points, and to define a workspace potential for each of these points. The global potential is obtained by summing the effects of the individual potential functions. Evaluating the effect a particular potential field on a single point is no different than the evaluations required in Section 7.2, but the required distance and gradient calculations are much simpler.

Let $\mathcal{A}_{\text{att}} = \{\boldsymbol{a}_1, \boldsymbol{a}_2 \cdots \boldsymbol{a}_n\}$ be a set of control points used to define the attractive potential fields. For an n-link arm, we could use the centers of mass for the n links, or the origins for the DH frames (excluding the fixed frame 0). We denote by $\boldsymbol{a}_i(\boldsymbol{q})$ the position of the i^{th} control point when the robot is at configuration $\boldsymbol{q}$. For each $\boldsymbol{a}_i \in \mathcal{A}_{\text{att}}$ we can define an attractive potential by

$$U_{\text{att}i}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\zeta_i ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})||^2 & : \ ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})|| \le d \\ \\ d\zeta_i ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})|| - \dfrac{1}{2}\zeta_i d^2 & : \ ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})|| > d \end{cases} . \tag{7.14}$$

For the single point $\boldsymbol{a}_i$, this function is analogous the attractive potential defined in Section 7.2; it combines the conic and quadratic potentials, and reaches its global minimum when the control point reaches its goal position in the workspace. If $\mathcal{A}_{\text{att}}$ contains a sufficient number of independent control points (the origins of the DH frames, e.g.), then when all control points reach their global minimum potential value, the configuration of the arm will be $\boldsymbol{q}_{\text{final}}$.

With this potential function, the workspace force for attractive control point $\boldsymbol{a}_i$ is defined by

$$\mathcal{F}_{\text{att},i}(q) = -\nabla U_{\text{att}i}(\boldsymbol{q}) \tag{7.15}$$

$$= \begin{cases} -\zeta_i(\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})) & : \ ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})|| \le d \\ \\ -\dfrac{d\zeta_i(\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}}))}{||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})||} & : \ ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})|| > d \end{cases} \tag{7.16}$$

For the workspace repulsive potential fields, we will select a set of fixed control points on the robot $\mathcal{A}_{\text{rep}} = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_m\}$, and define the repulsive potential for $\boldsymbol{a}_j$ as

$$U_{\text{rep}_j}(\boldsymbol{q}) = \begin{cases} \dfrac{1}{2}\eta_j \left(\dfrac{1}{\rho(\boldsymbol{a}_j(\boldsymbol{q}))} - \dfrac{1}{\rho_0} \right)^2 & : \ \rho(\boldsymbol{a}_j(\boldsymbol{q})) \le \rho_0 \\ \\ 0 & : \ \rho(\boldsymbol{a}_j(\boldsymbol{q})) > \rho_0 \end{cases} , \tag{7.17}$$

in which $\rho(\boldsymbol{a}_j(\boldsymbol{q}))$ is the shortest distance between the control point $\boldsymbol{a}_j$ and any *workspace* obstacle, and ρ_0 is the workspace distance of influence in the worksoace for obstacles. The negative gradient of each U_{rep_j} corresponds to a workspace repulsive force,

$$\mathcal{F}_{\mathrm{rep},j}(\boldsymbol{q}) = \begin{cases} \eta_j \left(\frac{1}{\rho(\boldsymbol{a}_j(\boldsymbol{q}))} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(\boldsymbol{a}_j(\boldsymbol{q}))} \nabla \rho(\boldsymbol{a}_j(\boldsymbol{q})) & : \ \rho(\boldsymbol{a}_j(\boldsymbol{q})) \leq \rho_0 \\ 0 & : \ \rho(\boldsymbol{a}_j(\boldsymbol{q})) > \rho_0 \end{cases}, \tag{7.18}$$

in which the notation $\nabla\rho(\boldsymbol{a}_j(\boldsymbol{q}))$ indicates the gradient $\nabla\rho(\boldsymbol{x})$ evaluated at $\boldsymbol{x} = \boldsymbol{a}_j(\boldsymbol{q})$. If $\boldsymbol{b}$ is the point on the workspace obstacle boundary that is closest to the repulsive control point $\boldsymbol{a}_j$, then $\rho(\boldsymbol{a}_j(\boldsymbol{q})) = ||\boldsymbol{a}_j(\boldsymbol{q}) - \boldsymbol{b}||$, and its gradient is

$$\nabla\rho(\boldsymbol{x})\Bigg|_{\boldsymbol{x}=\boldsymbol{a}_j(\boldsymbol{q})} = \frac{\boldsymbol{a}_j(\boldsymbol{q}) - \boldsymbol{b}}{||\boldsymbol{a}_j(\boldsymbol{q}) - \boldsymbol{b}||}, \tag{7.19}$$

i.e., the unit vector directed from $\boldsymbol{b}$ toward $\boldsymbol{a}_j(\boldsymbol{q})$.

It is important to note that this selection of repulsive control points does not guarantee that the robot cannot collide with an obstacle. Figure 7.5 shows an example where this is the case. In this figure, the repulsive control points $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are very far from the obstacle $\mathcal{O}$, and therefore the repulsive influence may not be great enough to prevent the robot edge E_1 from colliding with the obstacle. To cope with this problem, we can use a set of floating repulsive control points, $\boldsymbol{a}_{float,i}$, typically one per link of the robot arm. The floating control points are defined as points on the boundary of a link that are closest to any workspace obstacle. Obviously the choice of the $\boldsymbol{a}_{float,i}$ depends on the configuration $\boldsymbol{q}$. For the example shown in Figure 7.5, $\boldsymbol{a}_{float}$ would be located at the center of edge E_1, thus repelling the robot from the obstacle. The repulsive force acting on $\boldsymbol{a}_{float}$ is defined in the same way as for the other control points, using (7.18).

7.3.2 Mapping workspace forces to joint forces and torques

Above we have constrructed potential fields in the robot's workspace, and these fields induce artificial forces on the individual control points on the robot. In this section, we describe how these forces can be used to drive a gradient descent algorithm on the configuration space.

Suppose a force, $\mathcal{F}$ were applied to a point on the robot arm. Such a force would induce forces and torques on the robot's joints. If the joints did not resist these forces, a motion would occur. This is the key idea

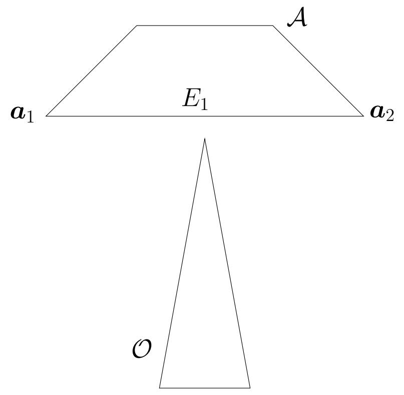


Figure 7.5: The repulsive forces exerted on the robot vertices $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ may not be sufficient to prevent a collision between edge E_1 and the obstacle.

behind mapping artificial forces in the workspace to motions of the robot arm. Therefore, we now derive the relationship between forces applied to the robot arm, and the resulting forces and torques that are induced on the robot joints.

Let F denote the vector of joint torques (for revolute joints) and forces (for prismatic joints) induced by the workspace force. As we will describe in Chapter 9, the principle of *virtual work* can be used to derive the relationship between $\mathcal{F}$ and F. Let $(\delta x, \delta y, \delta z)^T$ be a virtual displacement in the workspace and let $\delta \boldsymbol{q}$ be a virtual displacement of the robot's joints. Then, recalling that work is the inner product of force and displacement, by applying the principle of virtual work we obtain

$$\mathcal{F} \cdot (\delta x, \delta y, \delta z)^T = F \cdot \delta \boldsymbol{q} \tag{7.20}$$

which can be written as

$$\mathcal{F}^T (\delta x, \delta y, \delta z)^T = F^T \delta \boldsymbol{q}. \tag{7.21}$$

Now, recall from Chapter 5 that

$$\begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} = J \delta \boldsymbol{q},$$

in which J is the Jacobian of the forward kinematic map for linear velocity (i.e., the top three rows of the manipulator Jacobian). Substituting this into

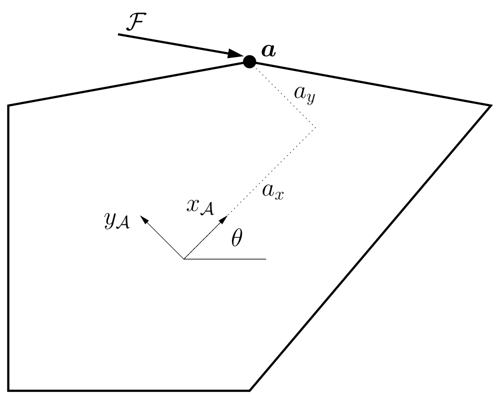


Figure 7.6: The robot $\mathcal{A}$, with coordinate frame oriented at angle θ from the world frame, and vertex $\boldsymbol{a}$ with local coordinates (a_x, a_y).

(7.20) we obtain

$$\mathcal{F}^T J\delta\boldsymbol{q} = F^T\delta\boldsymbol{q} \tag{7.22}$$

$$\tag{7.23}$$

and since this must hold for all virtual displacements $\delta\boldsymbol{q}$, we obtain

$$\mathcal{F}^T J = F^T \tag{7.24}$$

which implies that

$$J^T\mathcal{F} = F. \tag{7.25}$$

Thus we see that one can easily map forces in the workspace to joint forces and torques using (7.25).

Example 7.3 *A Force Acting on a Vertex of a Polygonal Robot.*

Consider the polygonal robot shown in Figure 7.6. The vertex $\boldsymbol{a}$ *has coordinates* $(a_x, a_y)^T$ *in the robot's local coordinate frame. Therefore, if the robot's configuration is given by* $\boldsymbol{q} = (x, y, \theta)$*, the forward kinematic map for vertex* $\boldsymbol{a}$ *(i.e., the mapping from* $\boldsymbol{q} = (x, y, \theta)$ *to the global coordinates of the vertex* $\boldsymbol{a}$*) is given by*

$$\boldsymbol{a}(x, y, \theta) = \left[\begin{array}{c} x + a_x\cos\theta - a_y\sin\theta \\ y + a_x\sin\theta + a_y\cos\theta \end{array}\right]. \tag{7.26}$$

The corresponding Jacobian matrix is given by

$$J_a(x, y, \theta) = \left[\begin{array}{ccc} 1 & 0 & -a_x\sin\theta - a_y\cos\theta \\ 0 & 1 & a_x\cos\theta - a_y\sin\theta \end{array}\right] \tag{7.27}$$

Therefore, the configuration space force is given by

$$\begin{aligned}\begin{bmatrix} F_x \\ F_y \\ F_\theta \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -a_x \sin\theta - a_y \cos\theta & a_x \cos\theta - a_y \sin\theta \end{bmatrix} \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \\ -\mathcal{F}_x(a_x \sin\theta - a_y \cos\theta) + \mathcal{F}_y(a_x \cos\theta - a_y \sin\theta) \end{bmatrix} \end{aligned} \tag{7.28}$$

and F_θ corresponds to the torque exerted about the origin of the robot frame.

In this simple case, one can use basic physics to arrive at the same result. In particular, recall that a force, $\mathcal{F}$, exerted at point, $\boldsymbol{a}$, produces a torque, τ, about the point $O_\mathcal{A}$, and this torque is given by the relationship $\tau = \boldsymbol{r} \times \mathcal{F}$, in which $\boldsymbol{r}$ is the vector from $O_\mathcal{A}$ to $\boldsymbol{a}$. Of course we must express all vectors relative to a common frame, and in three dimensions (since torque will be defined as a vector perpendicular to the plane in which the force acts). If we choose the world frame as our frame of reference, then we have

$$\boldsymbol{r} = \begin{bmatrix} a_x \cos\theta - a_y \sin\theta \\ a_x \sin\theta + a_y \cos\theta \\ 0 \end{bmatrix}$$

and the cross product gives

$$\begin{aligned}\tau &= \boldsymbol{r} \times \mathcal{F} \\ &= \begin{bmatrix} a_x \cos\theta - a_y \sin\theta \\ a_x \sin\theta + a_y \cos\theta \\ 0 \end{bmatrix} \times \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ -\mathcal{F}_x(a_x \sin\theta - a_y \cos\theta) + \mathcal{F}_y(a_x \cos\theta - a_y \sin\theta) \end{bmatrix} \end{aligned} \tag{7.29}$$

Thus we see that the more general expression $J^T\mathcal{F} = F$ gives the same value for torque as the expression $\tau = \boldsymbol{r} \times \mathcal{F}$ from mechanics.

◇

Example 7.4 *Two-link Planar Arm*

Consider a two-link planar arm with the usual DH frame assignment. If we assign the control points as the origins of the DH frames (excluding the base frame), the forward kinematic equations for the arm give

$$\begin{bmatrix} \boldsymbol{a}_1(\theta_1, \theta_2) & \boldsymbol{a}_2(\theta_1, \theta_2) \end{bmatrix} = \begin{bmatrix} l_1 \cos\theta_1 & l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 & l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{bmatrix}$$

in which l_i are the link lengths (we use l_i rather than a_i to avoid confusion of link lengths and control points). For the problem of motion planning, we require only the Jacobian that maps joint velocities to linear velocities,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_1 \end{bmatrix}. \tag{7.30}$$

For the two-link arm, The Jacobian matrix for $\boldsymbol{a}_2$ is merely the Jacobian that we derived in Chapter 5:

$$J_{\boldsymbol{a}_2}(\theta_1, \theta_2) \quad = \quad \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix}. \tag{7.31}$$

The Jacobian matrix for $\boldsymbol{a}_1$ is similar, but takes into account that motion of joint two does not affect the velocity of $\boldsymbol{a}_1$,

$$J_{\boldsymbol{a}_1}(\theta_1, \theta_2) \quad = \quad \begin{bmatrix} -a_1 s_1 & 0 \\ a_1 c_1 & 0 \end{bmatrix}. \tag{7.32}$$

◇

The total configuration space force acting on the robot is the sum of the configuration space forces that result from all attractive and repulsive control points

$$\begin{aligned} F(\boldsymbol{q}) &= \sum_i F_{\mathrm{att}i}(\boldsymbol{q}) + \sum_i F_{\mathrm{rep}_i}(\boldsymbol{q}) \\ &= \sum_i J_i^T(\boldsymbol{q}) \mathcal{F}_{\mathrm{att},i}(\boldsymbol{q}) + \sum_i J_i^T(\boldsymbol{q}) \mathcal{F}_{\mathrm{rep},i}(\boldsymbol{q}) \end{aligned} \tag{7.33}$$

in which $J_i(\boldsymbol{q})$ is the Jacobian matrix for control point $\boldsymbol{a}_i$. It is essential that the addition of forces be done in the configuration space and *not* in the workspace. For example, Figure 7.7 shows a case where two workspace forces, $\mathcal{F}_1$ and $\mathcal{F}_2$, act on opposite corners of a rectang. It is easy to see that $\mathcal{F}_1 + \mathcal{F}_2 = 0$, but that the combination of these forces produces a pure torque about the center of the square.

Example 7.5 *Two-link planar arm revisited. Consider again the two-link planar arm. Suppose that that the workspace repulsive forces are given by $\mathcal{F}_{\mathrm{rep},i}(\theta_1, \theta_2) = [\mathcal{F}_{x,i}, \mathcal{F}_{y,i}]^T$. For the two-link planar arm, the repulsive forces in the configuration space are then given by*

$$F_{\mathrm{rep}}(\boldsymbol{q}) = \begin{bmatrix} -a_1 s_1 & a_1 c_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathcal{F}_{x,1} \\ \mathcal{F}_{y,1} \end{bmatrix} \tag{7.34}$$

$$+ \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & a_1 c_1 + a_2 c_{12} \\ -a_2 s_{12} & a_2 c_{12} \end{bmatrix} \begin{bmatrix} \mathcal{F}_{x,2} \\ \mathcal{F}_{y,2} \end{bmatrix} \tag{7.35}$$

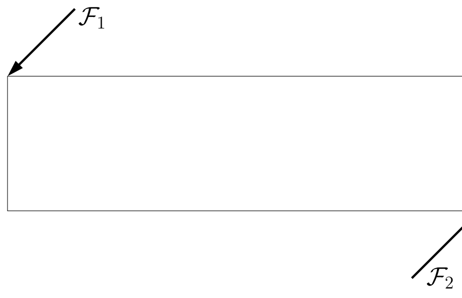


Figure 7.7: This example illustrates why forces must be mapped to the configuration space before they are added. The two forces illustrated in the figure are vectors of equal magnitude in opposite directions. Vector addition of these two forces produces zero net force, but there is a net moment induced by these forces.

◇

7.3.3 Motion Planning Algorithm

Having defined a configuration space force, we can use the same gradient descent method for this case as in Section 7.3. As before, there are a number of design choices that must be made.

ζ_i controls the relative influence of the attractive potential for control point $\boldsymbol{a}_i$. It is not necessary that all of the ζ_i be set to the same value. Typically, we weight one of the control points more heavily than the others, producing a "follow the leader" type of motion, in which the leader control point is quickly attracted to its final position, and the robot then reorients itself so that the other attractive control points reach their final positions.

η_j controls the relative influence of the repulsive potential for control point $\boldsymbol{a}_j$. As with the ζ_i it is not necessary that all of the η_j be set to the same value. In particular, we typically set the value of η_j to be much smaller for obstacles that are near the goal position of the robot (to avoid having these obstacles repel the robot from the goal).

ρ_0 As with the η_j, we can define a distinct ρ_0 for each obstacle. In particular, we do not want any obstacle's region of influence to include the goal position of any repulsive control point. We may also wish to assign distinct ρ_0's to the obstacles to avoid the possibility of overlapping regions of influence for distinct obstacles.

7.4 Using Random Motions to Escape Local Minima

As noted above, one problem that plagues artificial potential field methods for path planning is the existence of local minima in the potential field. In the case of articulated manipulators, the resultant field U is the sum of many attractive and repulsive fields defined over $\Re^3$. This problem has long been known in the optimization community, where probabilistic methods such as simulated annealing have been developed to cope with it. Similarly, the robot path planning community has developed what are known as *randomized methods* to deal with this and other problems. The first of these methods was developed specifically to cope with the problem of local minima in potential fields.

The first planner to use randomization to escape local minima was called RPP (for Randomized Potential Planner). The basic approach is straightforward: use gradient descent until the planner finds itself stuck in a local minimum, then use a random walk to escape the local minimum. The algorithm is a slight modification of the gradient descent algorithm of Section 7.3.

```
1.  $\boldsymbol{q}^0 \leftarrow \boldsymbol{q}_{\text{init}}$, $i \leftarrow 0$
2.  **IF** $\boldsymbol{q}^i \neq \boldsymbol{q}_{\text{final}}$
        $\boldsymbol{q}^{i+1} \leftarrow \boldsymbol{q}^i + \alpha^i \frac{F(\boldsymbol{q}^i)}{||F(\boldsymbol{q}^i)||}$
        $i \leftarrow i + 1$
    **ELSE** return $< \boldsymbol{q}^0, \boldsymbol{q}^1 \cdots \boldsymbol{q}^i >$
3.  **IF** stuck in a local minimum
        execute a random walk, ending at $\boldsymbol{q}'$
        $\boldsymbol{q}^{i+1} \leftarrow \boldsymbol{q}'$
4.  **GO TO** 2
```

The two new problems that must be solved are determining when the planner is stuck in a local minimum and defining the random walk. Typically, a heuristic is used to recognize a local minimum. For example, if several successive $\boldsymbol{q}^i$ lie within a small region of the configuration space, it is likely that there is a nearby local minimum (e.g., if for some small positive

ϵ we have $\|\boldsymbol{q}^i - \boldsymbol{q}^{i+1}\| < \epsilon$, $\|\boldsymbol{q}^i - \boldsymbol{q}^{i+2}\| < \epsilon$, and $\|\boldsymbol{q}^i - \boldsymbol{q}^{i+3}\| < \epsilon$ then assume $\boldsymbol{q}^i$ is near a local minimum).

Defining the random walk requires a bit more care. The original approach used in RPP is as follows. The random walk consists of t random steps. A random step from $\boldsymbol{q} = (q_1, \cdots q_n)$ is obtained by randomly adding a small fixed constant to each q_i,

$$\boldsymbol{q}_{\text{random}-\text{step}} = (q_1 \pm v_1, \cdots q_n \pm v_n)$$

with v_i a fixed small constant and the probability of adding $+v_i$ or $-v_i$ equal to 1/2 (i.e., a uniform distribution). Without loss of generality, assume that $\boldsymbol{q} = \mathbf{0}$. We can use probability theory to characterize the behavior of the random walk consisting of t random steps. In particular, the probability density function for $\boldsymbol{q}' = (q_1, \cdots, q_n)$ is given by

$$p_i(q_i, t) = \frac{1}{v_i\sqrt{2\pi t}} \exp\left(-\frac{q_i^2}{2v_i^2 t}\right) \tag{7.36}$$

which is a zero mean Gaussian density function with variance $v_i^2 t$. This is a result of the fact that the sum of a set of uniformly distributed random variables is a Gaussian random variable.[2] The variance $v_i^2 t$ essentially determines the range of the random walk. If certain characteristics of local minima (e.g., the size of the basin of attraction) are known in advance, these can be used to select the parameters v_i and t. Otherwise, they can be determined empirically, or based on weak assumptions about the potential field (the latter approach was used in the original RPP).

7.5 Probabilistic Roadmap Methods

The potential field approaches described above incrementally explore $\mathcal{Q}_{\text{free}}$, searching for a path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$. At termination, these planners return a single path. Thus, if multiple path planning problems must be solved, such a planner must be applied once for each problem. An alternative approach is to construct a representation of $\mathcal{Q}_{\text{free}}$ that can be used to quickly generate paths when new path planning problems arise. This is useful, for example, when a robot operates for a prolonged period in a single workspace.

[2] A Gaussian density function is the classical bell shaped curve. The mean indicates the center of the curve (the peak of the bell) and the variance indicates the width of the bell. The probability density function (pdf) tells how likely it is that the variable q_i will lie in a certain interval. The higher the pdf values, the more likely that q_i will lie in the corresponding interval.

In this section, we will describe probabilistic roadmaps (PRMs), which are one-dimensional roadmaps in $\mathcal{Q}_{\text{free}}$ that can be used to quickly generate paths. Once a PRM has been constructed, the path planning problem is reduced to finding paths to connect $\boldsymbol{q}_{\text{init}}$ and $\boldsymbol{q}_{\text{final}}$ to the roadmap (a problem that is typically much easier than finding a path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$).

A PRM is a network of simple curve segments, or arcs, that meet at nodes. Each node corresponds to a configuration. Each arc between two nodes corresponds to a collision free path between two configurations. Constructing a PRM is a conceptually straightforward process. First, a set of random configurations is generated to serve as the nodes in the network. Then, a simple, local path planner is used to generate paths that connect pairs of configurations. Finally, if the initial network consists of multiple connected components[3], it is augmented by an enhancement phase, in which new nodes and arcs are added in an attempt to connect disjoint components of the network. To solve a path planning problem, the simple, local planner is used to connect $\boldsymbol{q}_{\text{init}}$ and $\boldsymbol{q}_{\text{final}}$ to the roadmap, and the resulting network is searched for a path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$. These four steps are illustrated in Figure 7.8. We now discuss these steps in more detail.

7.5.1 Sampling the configuration space

The simplest way to generate sample configurations is to sample the configuration space uniformly at random. Sample configurations that lie in $\mathcal{QO}$ are discarded. A simple collision checking algorithm can determine when this is the case. The disadvantage of this approach is that the number of samples it places in any particular region of $\mathcal{Q}_{\text{free}}$ is proportional to the volume of the region. Therefore, uniform sampling is unlikely to place samples in narrow passages of $\mathcal{Q}_{\text{free}}$. In the PRM literature, this is refered to as the *narrow passage problem*. It can be dealt with either by using more intelligent sampling schemes, or by using an enhancement phase during the construction of the PRM. In this section, we discuss the latter option.

7.5.2 Connecting Pairs of Configurations

Given a set of nodes that correspond to configurations, the next step in building the PRM is to determine which pairs of nodes should be connected by a simple path. The typical approach is to attempt to connect each node to it's k nearest neighbors, with k a parameter chosen by the user. Of course,

[3] A connected component is a maximal subnetwork of the network such that a path exists in the subnetwork between any two nodes.

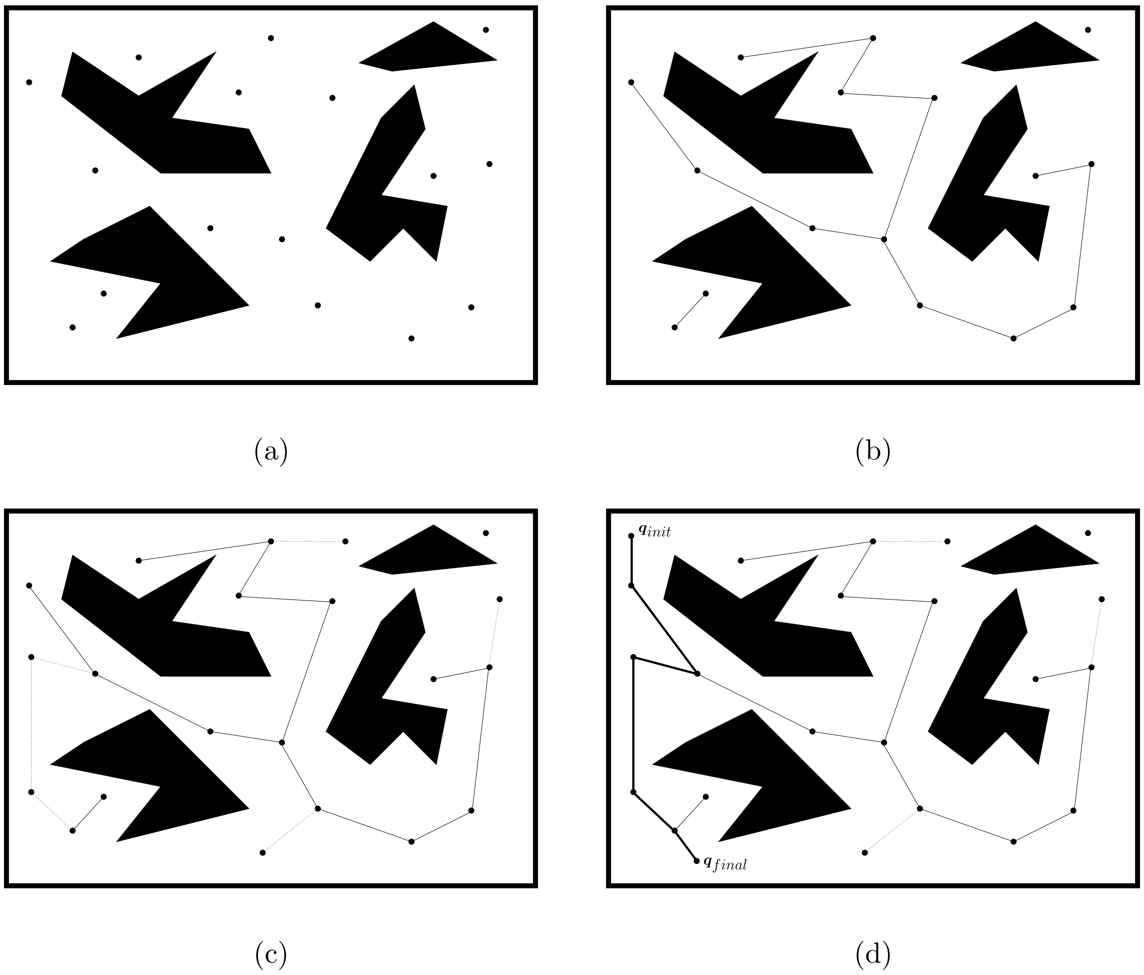


Figure 7.8: (a) A two-dimensional configuration space populated with several random samples (b) One possible PRM for the given configuration space and random samples (c) PRM after enhancement (d) path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$ found by connecting $\boldsymbol{q}_{\text{init}}$ and $\boldsymbol{q}_{\text{final}}$ to the roadmap and then searching the roadmap for a path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$.

2-norm in C-space:	$\lVert \boldsymbol{q}' - \boldsymbol{q} \rVert = \left[\sum_{i=1}^{n} (q_i' - q_i)^2 \right]^{\frac{1}{2}}$
∞-norm in C-space:	$\max_n \lvert q_i' - q_i \rvert$
2-norm in workspace:	$\left[\sum_{\mathrm{p} \in \mathcal{A}} \left\lVert \mathrm{p}(\boldsymbol{q}') - \mathrm{p}(\boldsymbol{q}) \right\rVert^2 \right]^{\frac{1}{2}}$
∞-norm in workspace:	$\max_{\mathrm{p} \in \mathcal{A}} \left\lVert \mathrm{p}(\boldsymbol{q}') - \mathrm{p}(\boldsymbol{q}) \right\rVert.$

Table 7.1: Four distance functions from the literature that we have investigated

to define the nearest neighbors, a distance function is required. Table 7.1 lists four distance functions that have been popular in the PRM literature. For the equations in this table, the robot has n joints, $\boldsymbol{q}$ and $\boldsymbol{q}'$ are the two configurations corresponding to different nodes in the roadmap, q_i refers to the configuration of the ith joint, and $\mathrm{p}(\boldsymbol{q})$ refers to the workspace reference point p of a set of reference points of the robot, $\mathcal{A}$, at configuration $\boldsymbol{q}$. Of these, the simplest, and perhaps most commonly used, is the 2-norm in configuraiton space.

Once pairs of neighboring nodes have been identified, a simple local planner is used to connect these nodes. Often, a straight line in configuration space is used as the candidate plan, and thus, planning the path between two nodes is reduced to collision checking along a straight line path in the configuration space. If a collision occurs on this path, it can be discarded, or a more sophisticated planner (e.g., RPP discussed above) can be used to attempt to connect the nodes.

The simplest approach to collision detection along the straight line path is to sample the path at a sufficiently fine discretization, and to check each sample for collision. This method works, provided the discretization is fine enough, but it is terribly inefficient. This is because many of the computations required to check for collision at one sample are repeated for the next sample (assuming that the robot has moved only a small amount between the two configurations). For this reason, incremental collision detection approaches have been developed. While these approaches are beyond the scope of this text, a number of collision detection software packages are available in the public domain. Most developers of robot motion planners use one of these packages, rather than implementing their own collision detection routines.

7.5.3 Enhancement

After the initial PRM has been constructed, it is likely that it will consist of multiple connected components. Often these individual components lie in large regions of $\mathcal{Q}_{\text{free}}$ that are connected by narrow passages in $\mathcal{Q}_{\text{free}}$. The goal of the enhancement process is to connect as many of these disjoint components as possible.

One approach to enhancement is to merely attempt to directly connect nodes in two disjoint components, perhaps by using a more sophisticated planner such as RPP. A common approach is to identify the largest connected component, and to attempt to connect the smaller components to it. The node in the smaller component that is closest to the larger component is typically chosen as the candidate for connection. A second approach is to choose a node randomly as candidate for connection, and to bias the random choice based on the number of neighbors of the node; a node with fewer neighbors in the network is more likely to be near a narrow passage, and should be a more likely candidate for connection.

A second approach to enhancement is to add samples more random nodes to the PRM, in the hope of finding nodes that lie in or near the narrow passages. One approach is to identify nodes that have few neighbors, and to generate sample configurations in regions around these nodes. The local planner is then used to attempt to connect these new configurations to the network.

7.5.4 Path Smoothing

After the PRM has been generated, path planning amounts to connecting $\boldsymbol{q}_{\text{init}}$ and $\boldsymbol{q}_{\text{final}}$ to the network using the local planner, and then performing path smoothing, since the resulting path will be composed of straight line segments in the configuration space. The simplest path smoothing algorithm is to select two random points on the path and try to connect them with the local planner. This process is repeated until until no significant progress is made.

7.6 Historical Perspective

The earliest work on robot planning was done in the late sixties and early seventies in a few University-based Artificial Intelligence (AI) labs [?, ?, ?]. This research dealt with high level planning using symbolic reasoning that was much in vogue at the time in the AI community. Geometry was not often explicitly considered in early robot planners, in part because it was not clear how to represent geometric constraints in a computationally plausible manner. The configuration space and its application to path planning were introduced in [?]. This was the first rigorous, formal treatment of the geometric path planning problem, and it initiated a surge in path planning research. The earliest work in geometric path planning developed methods to construct volumetric representations of the free configuration space. These included exact methods (e.g., [?]), and approximate methods (e.g., [?, ?, ?]). In the former case, the best known algorithms have exponential complexity and require exact descriptions of both the robot and its environment, while in the latter case, the size of the representation of C-space grows exponentially in the dimension of the C-space. The best known algorithm for the path planning problem, giving an upper bound on the amount of computation time required to solve the problem, appeared in [?]. That real robots rarely have an exact description of the environment, and a drive for faster planning systems led to the development of potential fields approaches [?, ?].

By the early nineties, a great deal of research had been done on the geometric path planning problem, and this work is nicely summarized in the textbook [?]. This textbook helped to generate a renewed interest in the path planning problem, and it provided a common framework in which to analyze and express path planning algorithms. Soon after, the research field of *Algorithmic Robotics* was born at a small workshop in San Francisco [?].

In the early nineties, randomization was introduced in the robot planning community [?], originally to circumvent the problems with local minima in potential fields). Early randomized motion planners proved effective for a large range of problems, but sometimes required extensive computation time for some robots in certain environments [?]. This limitation, together with the idea that a robot will operate in the same environment for a long period of time led to the development of the probabilistic roadmap planners [?, ?, ?].

Finally, much work has been done in the area of collision detection in recent years. [?, ?, ?, ?]. This work is primarily focused on finding efficient, incremental methods for detecting collisions between objects when one or both are moving. A number of public domain collision detection software packages are currently available on the internet.

7.7 Problems

1. Show that for $U(\boldsymbol{q}) = d\zeta\rho_f(\boldsymbol{q})$,

$$F_{\text{att}}(q) = -\nabla U(\boldsymbol{q}) = -\frac{d\zeta(\boldsymbol{q} - \boldsymbol{q}_{\text{final}})}{\rho_f(\boldsymbol{q})}.$$

2. Show that for

$$U_{\text{rep}}(\boldsymbol{q}) = \frac{1}{2}\eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)^2$$

   the repulsive force is given by

$$F_{\text{rep}}(\boldsymbol{q}) = -\nabla U_{\text{rep}}(\boldsymbol{q}) = \eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(\boldsymbol{q})}\nabla\rho(\boldsymbol{q}).$$

3. Implement an algorithm that computes $\mathcal{QO}$ for the case of a polygonal robot translating among polygonal obstacles.

4. Implement a potential field planner for the case of a polygon translating in the plane among polygonal obstacles. Use your solution to Problem 3 to compute $\mathcal{QO}$.

5. Implement a potential field planner for the case of a polygon that can translate and rotate in the plane among polygonal obstacles. Experiment with your choice of control points as well as the values for ζ_i, η_j and ρ_0.

6. Add a random walk step to your solution to problem 5 so that your robot can escape local minima in the potential field.

7. Implement a potential field planner for an planar arm with three degrees of freedom.

8. Download from the internet a collision detection package and experiment with it.

9. Implement a simple PRM planning algorithm for a multi-link planar arm.