axis. The size of $\mathcal{GC}$ is $O(r^m)$. We can represent the list OPEN as a balanced tree [Aho, Hopcroft and Ullman, 1983] so that each operation FIRST and INSERT takes logarithmic time in the size of OPEN, which is $O(r^m)$ in the worst case. The time complexity of the procedure BFP given above is thus $O(mr^m \log r)$.

Most of the time, the procedure BFP will only explore a small subset of $\mathcal{GC}$. Rather than representing $\mathcal{GC}$ explicitly in a large array and marking its configurations *visited* or *unvisited*, we may only represent the configurations that are installed in $T$. Then, instead of checking whether a configuration $\mathbf{q}'$ is marked *visited*, we test if it is in $T$. By storing the configurations in $T$ in a balanced-tree data structure sorted according to the configurations' coordinates, the test takes logarithmic time in the number of configurations in $T$, which is $O(r^m)$ in the worst case. This modification slightly increases the running time of the procedure (without changing its asymptotic time complexity), but reduces the amount of memory space it needs. It also makes it possible to run the algorithm with a grid $\mathcal{GC}$ of unbounded size (but then the running time is no longer bounded).

BFP is practical only when $m$ is small, say less than 5. In the case of a free-flying object in a two-dimensional workspace ($m = 3$), it provides a means for implementing a very fast and reliable planner with grid resolutions of the order of $256^3$ (see [Barraquand and Latombe, 1989a]). It may even be made faster (in average time) by using a pyramid of grids at various resolutions. However, when $m$ becomes larger, filling up local minimum wells is no longer tractable (the number of discretized configurations in a well tends to grow exponentially with the dimension $m$). Other techniques, e.g. the "randomized planning" technique presented in Section 5, must then be used.

### 3.3 Variational Planning

Another approach to path planning using a potential field consists of constructing a functional $J$ of a path $\tau$ and optimizing this functional over all possible paths. For example, if $\mathbf{U}$ is a potential function that is defined over the whole configuration space, with large values in the C-obstacle region and small values in free space, a possible definition of $J$ is the following:



$$J(\tau) = \int_0^1 \mathbf{U}(\tau(s))ds + \int_0^1 \|\frac{d\tau}{ds}\|ds.$$

The purpose of the first term in $J$ is to make the optimization produce a free path. The second term in $J$ is optional and is aimed at producing shorter paths. In fact, other objectives could be encoded in this form. The optimization of the functional $J$ can be done using standard variational calculus methods. We report the reader to publications describing implementations of this approach for more detail about the construction of both an appropriate potential $\mathbf{U}$ and an appropriate objective functional $J$, and about the optimization of this functional (e.g. [Buckley, 1985] [Gilbert and Johnson, 1985] [Hwang and Ahuja, 1988] [Suh and Shin, 1988]).

Variational path planning suffers from the same sort of drawback as depth-first planning. Indeed, since it essentially consists of minimizing a functional $J$ by following its negated gradient, it may get stuck at a local minimum of $J$ that does not correspond to a free path. In addition, the optimization of $J$ is conducted over a space of much larger dimension (the number of configurations in the path) and can be quite costly. The advantage of variational path planning is that it allows additional objective criteria to be encoded in $J$.

## 4 Other Potential Functions

A variety of potential functions other than those presented in Sections 1 and 2 have been proposed in the literature. The most interesting of them are aimed at either one of these two goals: (1) improving the "local dynamic behavior" of the robot along the generated paths; (2) reducing the number of local minima and/or the size of their attractive wells.

The first goal is important when the potential field method is used to generate a path on-line or in a depth-first fashion (although in the second case, the path can be improved in a postprocessing step) [Tilove, 1990]. One of the proposed function is the "generalized potential field" [Krogh, 1984] which is a function of *both* the robot's configuration and velocity. It is constructed in such a way that the robot is repelled by an obstacle only if it is close to the obstacle *and* its velocity points toward the obstacle. If the robot moves parallel to the obstacle, it does not have to be repulsed by it. Using a similar idea, Faverjon and Tournassoud [Faver-

jon and Tournassoud, 1987] described a planning method that consists of minimizing a quadratic criterion under linear constraints. The minimization pulls the robot toward the goal, while each constraint acts as a damper pushing the robot away from an obstacle when the robot is close to it and moves toward it. Such potential functions are interesting, but they address a secondary issue, the main one being the presence of local minima. They may even make it more difficult to deal with local minima.

In the following subsections we present theoretical and empirical results related to the definition of "good" potential functions with no, few, or small local minima[4]. The definition of the potential function is indeed a good place to start solving the local minima problem.

## 4.1 Notion of a Navigation Function

Since local minima other than the goal are a major cause of inefficiency for potential field methods, and since the potential function is not something imposed on us in the original formulation of the basic motion planning problem, an important question is the following: Is it possible to construct a potential function $\mathbf{U} : \mathcal{C}_{free} \rightarrow \mathbf{R}$ with a minimum located at $\mathbf{q}_{goal}$ whose domain of attraction includes the entire subset of $\mathcal{C}_{free}$ that is connected to $\mathbf{q}_{goal}$? Such a function, if it exists, is called a "global navigation function". If it could be constructed, then depth-first path generation would be guaranteed to reach $\mathbf{q}_{goal}$ whenever $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ belong to the same connected subset of $\mathcal{C}_{free}$, and to fail recognizably otherwise.

Using the Poincaré-Hopf Index Theorem [Guillemin and Pollack, 1974], Koditschek [Koditschek, 1987] showed that in the presence of C-obstacles (i.e. when $\mathcal{C}_{free}$ is strictly included in $\mathcal{C}$) a global navigation function does not exist in general. More specifically, if $\mathcal{C} = \mathbf{R}^2$ and if there are $q$ ($q > 0$) disjoint C-obstacles, each homeomorphic to the closed unit disc, then a potential function U must possess at least $q$ saddle points.

However, there seems to be no topological obstruction to the existence of

[4]Indeed, the number of local minima is not the only measure of the quality of a potential function. Another critical factor is the size of the attraction domain of these minima. A search technique may possibly escape from several small local minima without difficulty, but may fail to escape from a single, big one.



an analytical potential function with a minimum located at $\mathbf{q}_{goal}$ whose domain of attraction includes the whole subset of $\mathcal{C}_{free}$ that is connected to $\mathbf{q}_{goal}$, except a set of points of measure zero which are saddles of the potential function. These points are unstable equilibrium configurations, and any slight random deviation allows the planner to evade them. Such a potential function is called an "almost global navigation function". In the following, we will simply call it a **navigation function**. If $\mathcal{C} = \mathbf{R}^N$ and all the C-obstacles are disjoint spherical regions, then one can define a navigation function U using the formulae given in Section 1, as the sum of an attractive quadratic-well potential and repulsive potentials, each generated by a C-obstacle, with the distance of influence $\rho_0$ set to less than the minimum of half the minimal distance between any two C-obstacles and the distance between $\mathbf{q}_{goal}$ and the nearest C-obstacle.

Of course, the problem is to generalize this specific construct to C-obstacles with other shapes. One way to proceed is to construct a smooth "deformation" of these more complex C-obstacles into spherical ones. Indeed, if U is a navigation function over $\mathbf{R}^N$ and $\vartheta : \mathbf{R}^N \rightarrow \mathbf{R}^N$ is a diffeomorphism, then $\tilde{\mathbf{U}} = \mathbf{U} \circ \vartheta$ is also a navigation function [Rimon and Koditschek, 1988]. Hence, the problem is to construct a diffeomorphism $\vartheta$ that deforms a given set of C-obstacles into a set of spherical regions. Rimon and Koditschek [Rimon and Koditschek, 1989] constructed such a diffeomorphism in the particular case where the configuration space is a star-shaped subset[5] of $\mathbf{R}^N$ punctured by a finite number of disjoint $N$-dimensional star-shaped C-obstacles. They also extended the construction to C-obstacles of more complicated shapes defined as finite unions of intersecting star-shaped regions whose connectivity graph is a tree [Rimon and Koditschek, 1990].

## 4.2 Grid Potentials

Although it seems difficult to construct an analytical navigation function over a free space of arbitrary geometry, the computation of a numerical navigation function over a representation of the configuration space in the form of a grid turns out to be a much easier problem. Below we describe algorithms originally proposed in [Barraquand and Latombe,

[5]A subset $S$ of $\mathbf{R}^N$ is *star shaped* if and only if it is homeomorphic to the closed unit ball in $\mathbf{R}^N$ and there exists a point $z \in S$ such that for all points $p \in S$, the line segment $zp$ is contained in $S$.

1989a] for computing two such navigation functions (Subsections 4.2.1 and 4.2.2). These algorithms are efficient when the dimension of the configuration space is small, i.e. $m = 2$ and 3, and their time complexity is independent of the geometry of the free space. But they rapidly become impractical when $m$ grows larger. Nevertheless, when $m$ is large, they can still be used to construct potential fields with "good" properties (Subsection 4.2.3).

### 4.2.1 Simple Numerical Navigation Function

Let us discretize $\mathcal{C}$ into a rectangloid grid $\mathcal{GC}$ as we already did in Subsection 3.2. Each configuration in $\mathcal{GC}$ is labeled "0" if it lies in free space and "1" otherwise. The subset of configurations labeled "0" is denoted by $\mathcal{GC}_{free}$. We assume that both $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ belong to $\mathcal{GC}_{free}$.

A simple navigation function U is the $L^1$ distance (also called "Manhattan distance") to the goal in $\mathcal{GC}_{free}$. It can be easily computed using the following "wavefront expansion" algorithm [Barraquand, Langlois and Latombe, 1989b]. First, the value of U is set to 0 at $\mathbf{q}_{goal}$. Next, it is set to 1 at every 1-neighbor[6] of $\mathbf{q}_{goal}$ in $\mathcal{GC}$ (we assume that the distance between two 1-neighbors in $\mathcal{GC}$ is normalized to 1); to 2 at every 1-neighbor of these new configurations (if it has not been computed yet); etc. The algorithm terminates when the entire subset of $\mathcal{GC}_{free}$ accessible from $\mathbf{q}_{goal}$ has been fully explored.

Figures 3.1 and 3.2 illustrate the operations of the algorithms in a two-dimensional configuration space[7] represented by a $64^2$ grid[8]. The goal configuration $\mathbf{q}_{goal}$ is located in the upper left-hand corner of the grid. Figures 3.a through 3.g show various stages of the wavefront expansion. The displayed front lines are equipotential contours of the potential U computed by the algorithm. Figure 3.h shows a path following the steepest descent of U from an initial configuration $\mathbf{q}_{init}$. This path is a minimum-length path (for the $L^1$ distance in $\mathcal{GC}$) between $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$.

The algorithm computes U only in the connected subset of $\mathcal{GC}_{free}$ that

[6] Neighborhood in a grid is defined in Subsection 3.2.

[7] This space is treated here as a configuration space example. Later, we will also use the same space as a two-dimensional workspace example.

[8] This low resolution is intended to facilitate the interpretation of the figures.



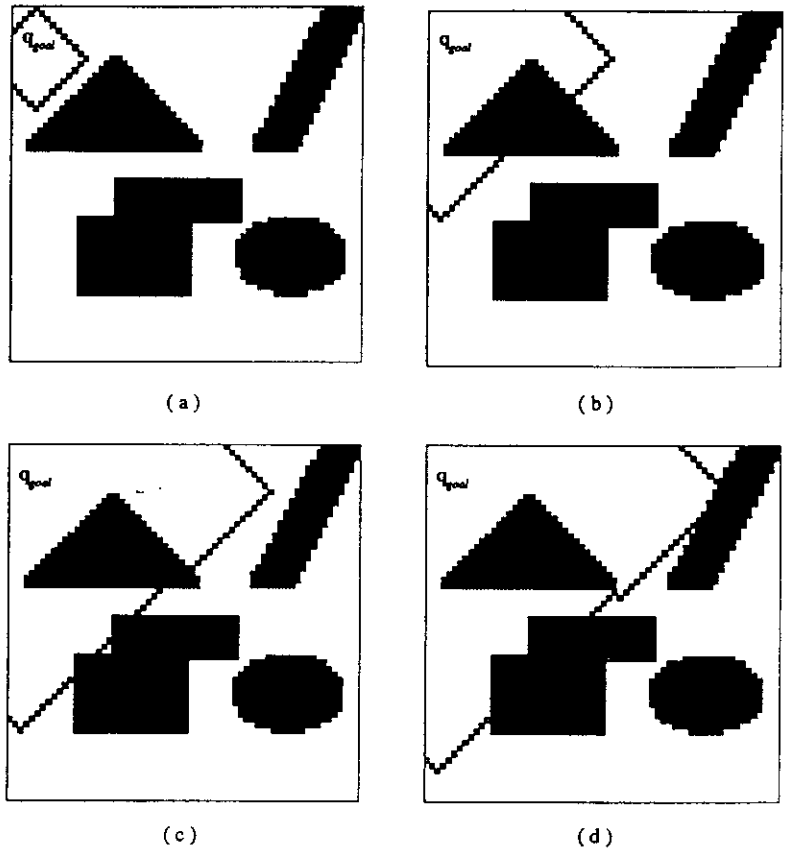



**Figure 3.1.** This figure and the next one illustrate the computation of a navigation function by the wavefront expansion procedure NF1 in a two-dimensional configuration space represented as a grid of $64^2$ configurations. Figures a through g show the expansion front at various stages of the computation. The displayed front lines are equipotential contours of the computed navigation function. Figure h displays a path following the steepest descent of the function from an initial configuration $\mathbf{q}_{init}$.

contains $\mathbf{q}_{goal}$. Hence, if $\mathrm{U}(\mathbf{q}_{init})$ has not been computed by the algorithm, we immediately know that no free path connects $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ at the resolution of the grid.

A more formal expression of the above algorithm is the following:

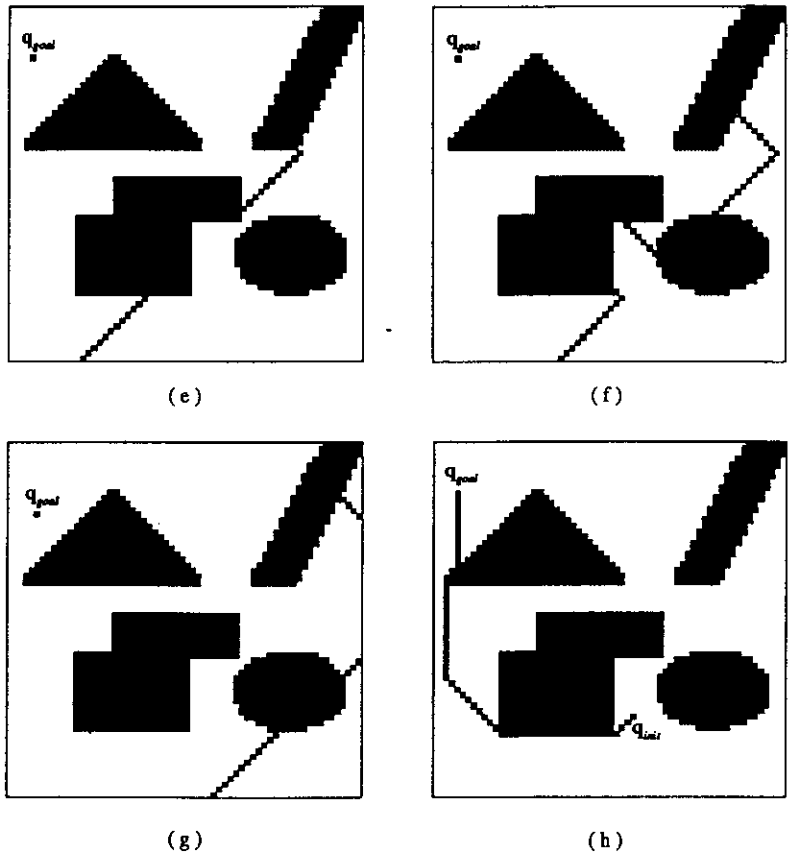



**Figure 3.2.** This figure is the continuation of Figure 3.1.

```
procedure NF1;
begin
    for every q in GC_free do U(q) ← M; [M is a large number]
    U(q_goal) ← 0; insert q_goal in L_0;
    [L_i, i = 0, 1, ..., is a list of configurations; it is initially empty]
    for i = 0, 1, ..., until L_i is empty do
        for every q in L_i do
            for every 1-neighbor q' of q in GC_free do
                if U(q') = M then
                    begin
                        U(q') ← i + 1;
                        insert q' at the end of L_{i+1};
                    end;
end;
```



The time complexity of this algorithm is linear in the number of configurations in the grid $\mathcal{GC}$. It is independent of the number and the shape of the C-obstacles.

An implemented planner using the navigation function computed by NF1 with a best-first search of the grid $\mathcal{GC}$ is described in [Lengyel et al., 1990] for a polygonal robot that both translates and rotates in $\mathcal{W} = \mathbf{R}^2$ among polygonal obstacles. The polygonal cross-sections at each discretized orientation in the grid $\mathcal{GC}$ are computed and passed to a rasterizing Computer Graphics hardware that gives the label "1" to the configurations of $\mathcal{GC}$ located in these cross-sections.

### 4.2.2 Improved Numerical Navigation Function

A drawback of the navigation function computed by the procedure NF1 given in the previous subsection is that it induces paths which in general graze C-obstacles. In contrast, the improved navigation function computed below induces paths that lie as far as possible away from the C-obstacles. But the $L^1$ lengths of these paths are no longer minimal.

The improved navigation function is computed in three steps. First, an $(m-1)$-dimensional subset $\mathcal{S}$ of $\mathcal{GC}_{free}$ called the **skeleton** is extracted. Second, the potential function U is computed in $\mathcal{S}$. Third, U is computed in the rest of $\mathcal{GC}_{free}$. We develop these three steps below:

**1. Computation of the skeleton.** The $L^1$ distance $d_1(\mathbf{q})$ from every configuration $\mathbf{q} \in \mathcal{GC}_{free}$ to the C-obstacle region is computed using a wavefront expansion algorithm with the expansion starting from the boundary of $\mathcal{GC}_{free}$. This boundary is constructed in lines 5-10 of the procedure L1 given below.

The following procedure computes the map $d_1$ over $\mathcal{GC}_{free}$:

```
1   procedure L1;
2   begin
```

```
3       for every q in GC_free do U(q) ← M; [M is a large number]
4       [L_i, i = 0,1,..., is a list of configurations; it is initially empty]
5       for every q in GC\GC_free do
6           if there exists a 1-neighbor q' of q in GC_free then
7               begin
8                   d_1(q) ← 0;
9                   insert q at the end of L_0;
10              end;
11      for i = 0,1,..., until L_i is empty do
12          for every q in L_i do
13              for every 1-neighbor q' of q in GC_free do
14                  if d_1(q') = M then
15                      begin
16                          d_1(q') ← i + 1;
17                          insert q' at the end of L_{i+1};
18                      end;
19  end;
```

Concurrently with the computation of the map $d_1$, an $(m-1)$-dimensional subset[9] $\mathcal{S}$ of $\mathcal{GC}_{free}$ is constructed as the set of configurations where the "waves" (the lists $L_i$ in the above procedure) issued from the boundary of $\mathcal{GC}_{free}$ meet. This is done by propagating not only the values of $d_1$, but also the points in the discretized boundary of $\mathcal{GC}_{free}$ that are at the origin of the propagation. For each configuration $\mathbf{q}$ in $\mathcal{GC}_{free}$ such that $d_1(\mathbf{q})$ has been computed, let $O(\mathbf{q})$ denote the point computed as follows. At line 8 of the algorithm, we set $O(\mathbf{q})$ to $\mathbf{q}$. At line 16, we set $O(\mathbf{q}')$ to $O(\mathbf{q})$. The "if" statement at lines 14-18 is extended by an "else" part that considers the case where $d_1(\mathbf{q}') \neq \mathrm{M}$, i.e. the case where two waves possibly meet, and builds $\mathcal{S}$: if the $L^1$ distance $\mathrm{D1}(O(\mathbf{q}'), O(\mathbf{q}))$ between $O(\mathbf{q}')$ and $O(\mathbf{q})$ is greater than some threshold $\alpha$, we include $\mathbf{q}$ in $\mathcal{S}$.

The resulting complete algorithm which computes both the map $d_1$ and the skeleton $\mathcal{S}$ is the following (the set $\mathcal{S}$ is initially empty):

```
procedure SKELETON;
begin
    for every q in GC_free do U(q) ← M; [M is a large number]
    [L_i, i = 0,1,..., is a list of configurations; it is initially empty]
    for every q in GC\GC_free do
        if there exists a 1-neighbor q' of q in GC_free then
            begin
                d_1(q) ← 0; O(q) = q;
                insert q at the end of L_0;
            end;
    for i = 0,1,..., until L_i is empty do
        for every q in L_i do
            for every 1-neighbor q' of q in GC_free do
                if d_1(q') = M then
                    begin
                        d_1(q') ← i + 1; O(q') ← O(q);
                        insert q' at the end of L_{i+1};
                    end;
                else if D1(O(q'), O(q)) > α then
                [α is an integer constant set to a value typically
                comprised between 2 and 6]
                    if q ∉ S then insert q' in S;
                    [this test avoids generating a double skeleton (i.e. a
                    skeleton of thickness 2) resulting from the fact that
                    two waves are meeting]
end;
```

[9] Actually, $\mathcal{S}$ is a discretized approximation of an $(m-1)$-dimensional subset of $\mathcal{C}$.



Figure 4 displays the skeleton computed in the same two-dimensional space as in Figure 3. In two dimensions, the skeleton is a kind of Voronoi diagram (see Section 2 of Chapter 4) for the $L^1$ distance. It is also similar to the "skeleton" extracted from a region in a digitized image using techniques from Mathematical Morphology [Serra, 1982].

**2. Computation of U in the skeleton.** The goal configuration $\mathbf{q}_{goal}$ is first connected to $\mathcal{S}$ by a path $\sigma$ following the steepest ascent of the map $d_1$ in $\mathcal{CG}_{free}$. This path $\sigma$ is included in $\mathcal{S}$ (lines 6-12 of the procedure NF2 given below). Then, U is computed in $\mathcal{S}$ by a wavefront expansion algorithm restricted to $\mathcal{S}$ and starting at $\mathbf{q}_{goal}$. The algorithm uses the $d_1$ map previously computed in order to guide the expansion. U is first given the value 0 at $\mathbf{q}_{goal}$, and $\mathbf{q}_{goal}$ is inserted in a list $Q$ of configuration sorted by decreasing value of $d_1$. Next, until $Q$ is empty, the first element

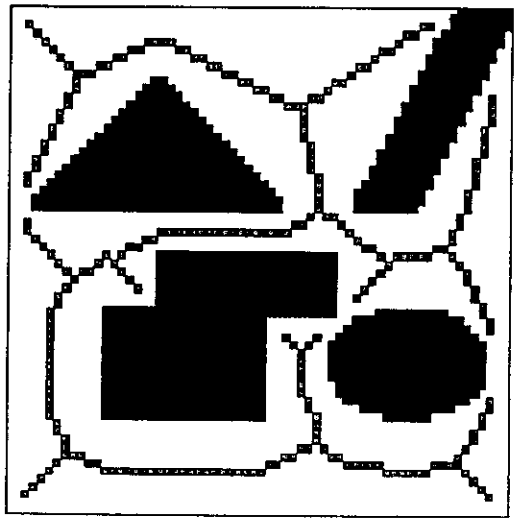


**Figure 4.** This figure shows the skeleton computed in the same two-dimensional space as in Figure 3 (with the parameter $\alpha$ equal to 4).

of $Q$ — call it $\mathbf{q}$ — is removed from $Q$; every $m$-neighbor[10] $\mathbf{q}'$ of $\mathbf{q}$ in $\mathcal{S}$ whose potential has not been computed yet receives a potential value equal to $\mathbf{U}(\mathbf{q}) + 1$ and is inserted in $Q$. The algorithm terminates when $Q$ is empty, i.e. when all the configurations in $\mathcal{S}$ accessible from $\mathbf{q}_{goal}$ have been given a potential value. A formal expression of the algorithm computing $\mathbf{U}$ in $\mathcal{S}$ consists of the lines 14-29 of the procedure NF2 given below. The outcome of the algorithm is a list $L_0$ of all the configurations of $\mathcal{S}$ accessible from $\mathbf{q}_{goal}$.

Figure 5 illustrates various stages of the wavefront propagation in the skeleton of Figure 4. The elements of $\mathcal{S}$ shown black are those which have been attained so far. $\mathcal{S}$ includes the path $\sigma$ that connects $\mathbf{q}_{goal}$ to the original skeleton.

**3. Computation of U in the rest of free space.** The potential $\mathbf{U}$ in the rest of the subset of $\mathcal{GC}_{free}$ accessible from $\mathbf{q}_{goal}$ is computed

[10] $\mathcal{S}$ is a subset of $\mathcal{GC}$ of dimension lower than $m$. In order to track it reliably, we must use the $m$-neighborhood.



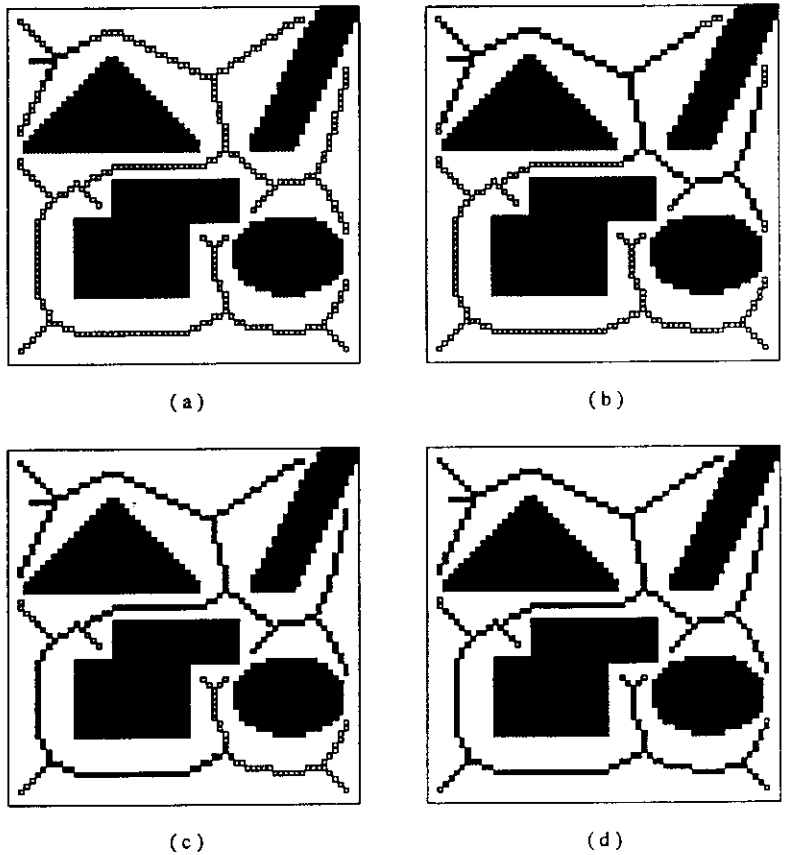
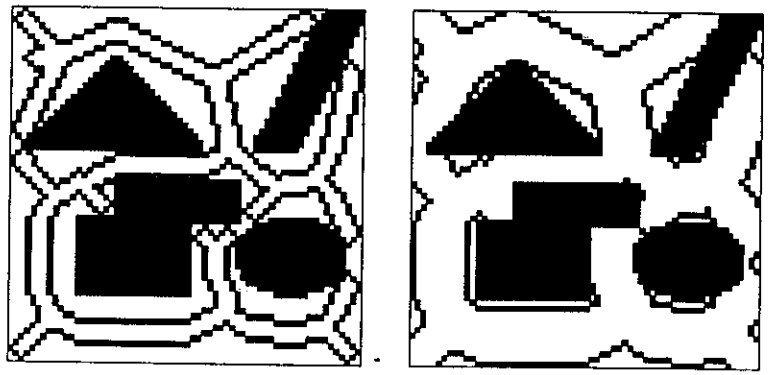



**Figure 5.** This figure illustrates four stages of the wavefront expansion carried out by the procedure NF2 in the skeleton of Figure 4. The skeleton elements shown black are those attained at the current stage of the propagation.

by a wavefront expansion algorithm starting from the configurations in $L_0$. The potential at each 1-neighbor $\mathbf{q}'$ of every configuration $\mathbf{q}$ in $L_0$ is set to $\mathbf{U}(\mathbf{q}) + 1$. Next, the potential at every neighbor $\mathbf{q}''$ of a configuration $\mathbf{q}'$ whose potential has been previously computed is set to $\mathbf{U}(\mathbf{q}') + 1$, etc. The algorithm (lines 34-41 of NF2) terminates when all the configurations of $\mathcal{GC}_{free}$ accessible from $\mathbf{q}_{goal}$ have been considered.

Figure 6 shows two stages of the wavefront expansion from the skeleton



(a) (b)

**Figure 6.** This figure shows two successive stages of the wavefront expansion carried out by the procedure NF2 from the skeleton of Figure 4.

of Figure 4.

The procedure NF2 given below is a more formal expression of the algorithm computing U in the subset of $\mathcal{GC}_{free}$ accessible from $\mathbf{q}_{goal}$:

```
1  procedure NF2;
2  begin
3      for every q in GC_free do U(q) ← M; [M is a large number]
4      [the following sequence of instructions connects q_goal to S]
5      [σ is a list of configurations; it is initially empty]
6      insert q_goal in σ; q ← q_goal;
7      while q ∉ S do
8          begin
9              select a neighbor q' having the largest value of d_1;
10             insert q' in σ; q ← q';
11         end;
12     S ← σ ∪ S;
13     [the following sequence of instructions computes U in S]
14     U(q_goal) ← 0; INSERT(q_goal, Q);
15     [Q is a list of configurations sorted by decreasing values of d_1;
16     it supports the operations FIRST, INSERT, and EMPTY (see
17     Subsection 3.2); it is initially empty]
```



```
18     [L_i, i = 0, 1, ..., is a list of configurations; it is initially empty]
19     while ¬ EMPTY(Q) do
20         begin
21             q ← FIRST(Q);
22             insert q at the end of L_0;
23             for every m-neighbors q' of q in S do
24                 if U(q') = M then
25                     begin
26                         U(q') ← U(q) + 1;
27                         INSERT(q', Q);
28                     end;
29         end;
30     [at the end of this loop, L_0 contains all the configurations in S
31     accessible from q_goal]
32     [the following instructions compute U in the rest of the subset
33     of GC_free accessible from q_goal]
34     for i = 0, 1, ..., until L_i is empty do
35         for every q in L_i do
36             for every neighbor q' of q in GC_free do
37                 if U(q') = M then
38                     begin
39                         U(q') ← U(q) + 1;
40                         insert q' at the end of L_{i+1};
41                     end;
42 end;
```

Figure 7 shows the potential function computed by NF2 in the two-dimensional space introduced in Figure 3. Figure 7.a displays equipotential contours of the function and Figures 7.b and c show three-dimensional views of the function. The valleys of the function are the skeleton $\mathcal{S}$ shown in Figure 4. This potential generates no stable equilibrium configuration other than $\mathbf{q}_{goal}$. However, it is usually not continuous and it is not suitable to explicitly compute its gradient. A best-first search (see Subsection 3.2) starting from any initial configuration $\mathbf{q}_{init}$ produces a path that first connects $\mathbf{q}_{init}$ to $\mathcal{S}$, then follows the curve in $\mathcal{S}$ that lies the furthest away from the C-obstacles, and finally connects $\mathcal{S}$ to $\mathbf{q}_{goal}$ (see Figure 7.d).

The time complexity of the above algorithm is slightly higher than that

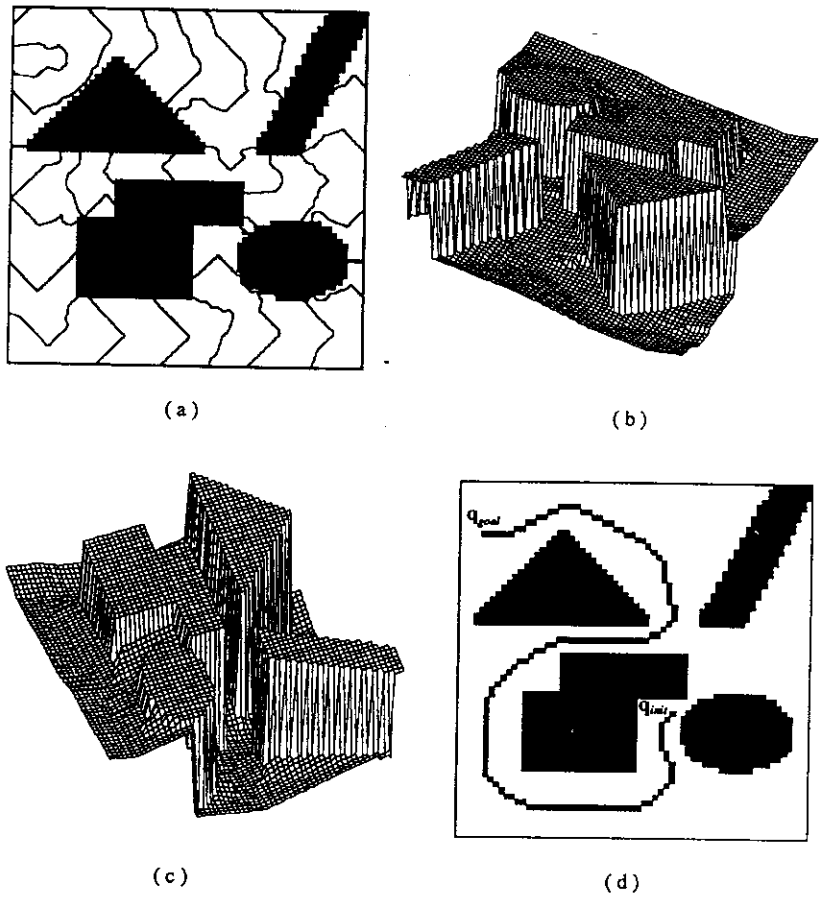



**Figure 7.** This figure illustrates the potential function **U** computed by the procedure NF2 in the same two-dimensional space as in Figure 3. Figure a displays equipotential contours of **U**. Figures b and c show two three-dimensional views of the function. Figure d shows a path generated by a best-first search of the grid starting at an initial configuration $\mathbf{q}_{init}$.

of the algorithm NF1 (Subsection 4.2.1). Let $n$ be the number of configurations in $\mathcal{GC}$ and $s$ (resp. $s'$) the number of configurations in $\mathcal{S}$ (resp. in $\mathcal{S}$ augmented by $\sigma$). By representing $\mathcal{S}$ in a balanced tree, the complexity of SKELETON is $O(n + s \log s)$. By representing the list $Q$ in a balanced tree, the complexity of NF2 is $O(n + s' \log s')$. In general,



we can expect $s$ and $s'$ to be proportional to $n^{\frac{m-1}{m}}$ since $\mathcal{S}$ and $\sigma$ are $(m-1)$-dimensional subsets of $\mathcal{GC}$. Under this condition, the complexity of both SKELETON and NF2 is $O(n + n^{\frac{m-1}{m}} \log n)$, and hence is linear in $n$.

Variants of the above potential are easy to define. For example, at line 39 of NF2, we could compute $\mathbf{U}(\mathbf{q}') = \mathbf{U}(\mathbf{q}) + 1/d_1(\mathbf{q})$, which yields a potential **U** that becomes infinite on the boundary of $\mathcal{CG}_{free}$.

### 4.2.3 Application to High-Dimensional Spaces

The procedures NF1 and NF2 are general in theory. However, in practice, they can only be applied to configuration spaces of small dimension. Indeed, the size of the grid $\mathcal{GC}$ increases exponentially with the dimension $m$ of the robot's configuration space. Hence, both NF1 and NF2 require exponential space and time in $m$. Nevertheless, in high-dimensional spaces, we can apply a technique similar to the one presented in Subsections 2.2 and 2.3. It consists of using NF1 and NF2 to compute "workspace potentials" associated with *control points* in the robot and combining them into a "configuration space potential", as described below. Since the workspace dimension is 2 or 3, the cost of running NF1 or NF2 remains reasonable.

Let $\mathcal{GW}$ be a regular grid of points thrown across $\mathcal{A}$'s workspace $\mathcal{W}$. This grid is constructed by discretizing the axes of the frame $\mathcal{F}_{\mathcal{W}}$ embedded in $\mathcal{W}$. We assume that the grid is bounded and forms a rectangloid. Each point in $\mathcal{GW}$ is labeled "1" if it lies in the obstacle region $\mathcal{B}$ or if it lies in the boundary of $\mathcal{GW}$; it is labeled "0" otherwise. $\mathcal{GW}_{free}$ denotes the subset of points of $\mathcal{GW}$ that are labeled "0".

Let $a_j$, $j = 1, ..., Q$, be the control points selected in $\mathcal{A}$. With each point $a_j$ we associate a potential function $\mathbf{V}_j$ defined over $\mathcal{GW}_{free}$. $\mathbf{V}_j$ is computed using either NF1 or NF2, by substituting $\mathcal{GW}$, $\mathcal{GW}_{free}$, $a_j(\mathbf{q}_{goal})$, $\mathbf{x}$, and $\mathbf{x}'$ for $\mathcal{GC}$, $\mathcal{GC}_{free}$, $\mathbf{q}_{goal}$, $\mathbf{q}$, and $\mathbf{q}'$, respectively, in the text of the procedures.

Once the potentials $\mathbf{V}_j$, $j = 1, ..., Q$, have been computed over $\mathcal{GW}_{free}$, the potential function **U** is computed at a configuration $\mathbf{q} \in \mathcal{CG}_{free}$ as:

$$\mathbf{U}(\mathbf{q}) = G(\mathbf{V}_1(a_1(\mathbf{q})), ..., \mathbf{V}_Q(a_Q(\mathbf{q})))$$

where $G$ is called the **arbitration function** (we will give examples of $G$ below). This potential function is not computed at every configuration in $\mathcal{GC}$ (as mentioned above, this would be impractical), but only at the configurations where the planner needs to know the value of $\mathbf{U}$ (this is in general a very small subset of $\mathcal{GC}$ relative to the total size of the grid). In order to compute $\mathbf{U}$ at a given configuration $\mathbf{q}$, the position $a_j(\mathbf{q})$ of every control point is first computed and the value of $\mathbf{V}_j$ at the closest position of $a_j(\mathbf{q})$ in $\mathcal{GW}$ is retrieved and used to compute $\mathbf{U}(\mathbf{q})$. With this definition, $\mathbf{U}$ is positive or null over free space and becomes null at the goal configurations. If $Q = N$, the goal configuration is usually uniquely defined. If $Q \leq N$, there are multiple goal configurations.

The above computation requires that the resolutions of the grids $\mathcal{GW}$ and $\mathcal{GC}$ be related to each other appropriately. Let $\Delta_i$, $i = 1, ..., m$, be the increment between two discretization points along the $i^{\text{th}}$ axis of $\mathcal{GC}$. We assume that the scales along the $m$ axes of $\mathcal{GC}$ have been normalized so that a change of the robot's configuration by $\Delta_i$ along one axis of $\mathcal{GC}$ causes the longest Euclidean displacement of the points in $\mathcal{A}$ to be of the same order of magnitude as the one caused by a change by $\Delta_j$ along another axis (see Subsection 3.2). Let $\delta$ be the increment between two discretization points along an axis of the grid $\mathcal{GW}$. In the same way as the $\Delta_i$'s have been normalized, the increment $\delta$ should be of the same order of magnitude as the length of the longest Euclidean displacement of the points in $\mathcal{A}$ when its configuration is changed by $\Delta_i$ along any of its axes. Hence, $\delta$ is equal to the increment $\Delta_i$ along any of the translational axes of $\mathcal{GC}$.

The rationale behind the two-step construction of $\mathbf{U}$ is to avoid computing $\mathbf{U}$ systematically over an enormous grid, while using the workspace as a source of inspiration for generating a "good" function $\mathbf{U}$. The first goal is achieved since the only potentials which we exhaustively compute are the workspace potentials. The second goal is hopefully achieved by combining workspace potentials that are free of local minima. Their combination certainly allows us to construct a potential $\mathbf{U}$ that prevents the robot from getting trapped in simple concavities formed by the obstacles. However, the combination usually have local minima. Typically, these minima result from the fact that the various control points, which have fixed relative positions, are concurrently attracted toward their respective goal positions. Hence, they are competing among themselves to



attain their goal positions. This competition creates unwanted minima and the role of the arbitration function $G$ is to arbitrate in this competition. Various arbitration functions are possible, resulting in more or less numerous, more or less deep local minima.

A typical arbitration function consists of adding the various potentials $\mathbf{V}_j$, i.e. :

$$\mathbf{U}(\mathbf{q}) = \sum_{j=1}^{j=Q} \mathbf{V}_j(a_j(\mathbf{q})).$$

However, this choice does not favor any control point over any other. For this reason, it tends to increase the number of conflicts and to produce numerous minima.

Another choice for $G$ is:

$$\mathbf{U}(\mathbf{q}) = \min_{j=1}^{j=Q} \mathbf{V}_j(a_j(\mathbf{q})) + \varepsilon \max_{j=1}^{j=Q} \mathbf{V}_j(a_j(\mathbf{q})) \quad (1)$$

where $\varepsilon$ is a small constant (typically, $\varepsilon = 0.1$). This arbitration function favors the attraction of the point in $\mathcal{A}$ that is the closest to its goal position. The second term of $G$ makes the robot rotate (less vigorously) toward a goal configuration. Experience shows that this arbitration function tends to generate fewer local minima than the previous one. However, it may create a deep local minimum if there is not enough space for the robot to rotate when it is close to its goal position.

Another choice for $G$ is:

$$\mathbf{U}(\mathbf{q}) = \max_{j=1}^{j=Q} \mathbf{V}_j(a_j(\mathbf{q})).$$

This choice tends to increase the number of competitions among the control points and, therefore, the number of local minima. But experimental results show that it also tends to reduce the size of the attraction domains of these minima.

In general, NF2 computes workspace potentials $\mathbf{V}_j$ that can be combined into a "better" potential function $\mathbf{U}$ than those computed by NF1. This results from the fact that, unlike NF1, NF2 tends to keep the control points as far away as possible from the obstacles. Hence, it increases the maneuvering space for the robot.

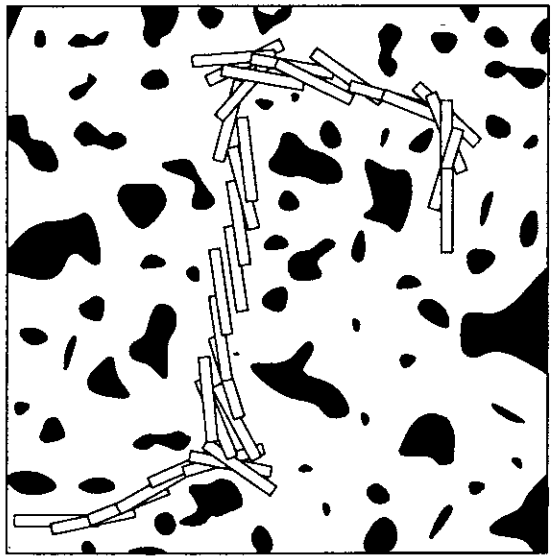


**Figure 8.** This figure illustrates a path for a rectangular robot moving among many obstacles. The path has been generated by a best-first planner using a potential field defined by equation (1) with two potentials $\mathbf{V}_j$ computed by NF2 for two control points located at the small ends of the robot.

The potential U defined in equation (1) has been used in a very efficient path planner using a best-first search technique (see Subsection 3.2) for a robot that can both translate and rotate in the plane [Barraquand and Latombe, 1989a]. Since the function U does not grow to infinity near the boundary of the C-obstacle region, best-first search using this function does not guarantee avoiding collision with obstacles. Collisions have to be checked separately in the workspace (as will be explained in Subsection 5.2). Figure 8 shows a path generated by this planner with two control points located at the small ends of the robot.

## 4.3 Elliptical Potentials

Another approach to reducing the number and the size of the local minima of the potential function U consists of defining U as the combination of an attractive potential having a revolute symmetry around the goal configuration $\mathbf{q}_{goal}$ (e.g. a parabolic well) and elliptical repulsive



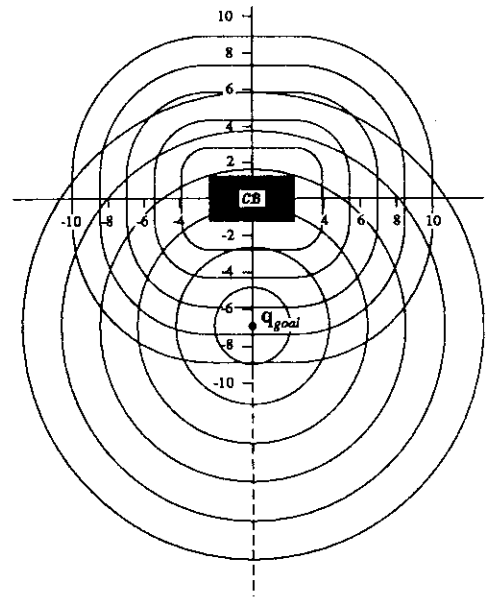



**Figure 9.** The equipotential contours of the attractive potential field (parabolic well) are circles centered at $\mathbf{q}_{goal}$. The equipotential contours of the repulsive potential field are of the form $\max_{\mathbf{q}' \in \mathcal{CB}} \|\mathbf{q} - \mathbf{q}'\| = K$, with $K$ constant. They are "rectangles with circular corners".

potentials. The basic idea is to define the repulsive potential around a C-obstacle in such a way that the equipotential contours converge toward the boundary of the C-obstacle when the distance to the C-obstacle tends toward zero, and toward spherical surfaces when the distance increases [Volpe and Khosla, 1987]. We develop this idea in a two-dimensional configuration space $\mathcal{C}$.

Consider the case where there is a single rectangular C-obstacle $\mathcal{CB}$ in $\mathcal{C}$ and the goal configuration $\mathbf{q}_{goal}$ is located on a symmetry axis of $\mathcal{CB}$ (Figure 9). Let the potential in $\mathcal{C}_{free}$ be the function $\mathbf{U} = \mathbf{U}_{att} + \mathbf{U}_{rep}$ defined in Section 1. The equipotential contours of $\mathbf{U}_{att}$ are circles centered at $\mathbf{q}_{goal}$. Within the influence distance $\rho_0$ from $\mathcal{CB}$, the equipotential contours of $\mathbf{U}_{rep}$ are "rectangles with circular corners". Figure 10 illustrates how the composition of $\mathbf{U}_{att}$ and $\mathbf{U}_{rep}$ creates a local minimum of U along the symmetry axis of $\mathcal{CB}$ containing $\mathbf{q}_{goal}$:

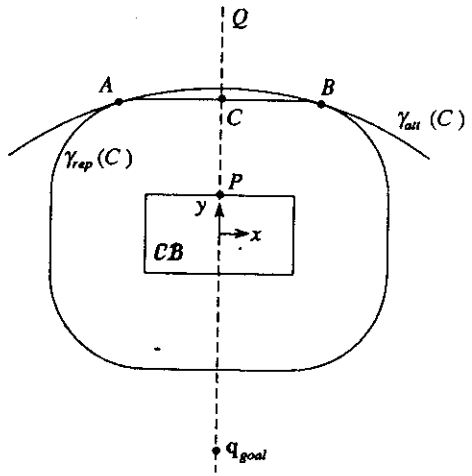



**Figure 10.** A local minimum is located on the symmetry axis of $\mathcal{CB}$, which contains $q_{goal}$, "uphill" from $\mathcal{CB}$ with respect to $q_{goal}$. More generally, a local minimum may appear whenever the radius of curvature of a repulsive equipotential curve at a configuration is greater than the radius of curvature of the attractive equipotential curve at the same configuration [Volpe and Khosla, 1987].

- Consider the half-line $PQ$. $P$ is on the boundary of $\mathcal{CB}$, while $Q$ lies at infinity. Let $\mathbf{q}$ be a configuration in $PQ$ at a finite non-null distance from $P$. The potential $\mathbf{U}(\mathbf{q})$ is finite. When $\mathbf{q}$ tends toward either $P$ or $Q$, $\mathbf{U}(\mathbf{q})$ tends toward $+\infty$.

- Let $C$ be a point in $PQ$ at a finite non-null distance from $P$, $\gamma_{rep}(C)$ be the equipotential contour of $\mathbf{U}_{rep}$ that contains $C$, and $\gamma_{att}(C)$ be the equipotential contour of $\mathbf{U}_{att}$ that is tangent to $\gamma_{rep}(C)$ at two distinct points $A$ and $B$. As $\mathbf{q}$ moves along $\gamma_{rep}(C)$ from $A$ to $B$, the potential $\mathbf{U}$ first decreases until $C$ is reached, then increases until $B$ is reached.

Therefore, the potential function $\mathbf{U}$ has a local minimum at a configuration $\mathbf{q}_{loc}$ located on $PQ$ between $P$ and $Q$. This minimum results from the fact that the radius of curvature of the repulsive equipotential curve at $\mathbf{q}_{loc}$ is greater than the radius of curvature of the attractive equipotential at $\mathbf{q}_{loc}$. For this reason, the total potential is non-convex despite the fact that both the attractive potential and the repulsive potential



are convex.

This remark suggests that we should approximate the C-obstacle region by a collection of disjoint discs. If $\mathbf{q}_{goal}$ is outside any of these discs, we can select the distance of influence $\rho_0$ to be smaller than the minimum of half the minimal distance between any two discs and the distance between $\mathbf{q}_{goal}$ and the nearest disc. Then, it is guaranteed that the radius of the repulsive equipotential curve at any configuration $\mathbf{q}$ is smaller than the radius of the attractive equipotential at $\mathbf{q}$. The total potential is a navigation function previously proposed in Subsection 4.1. However, the drawback of approximating the C-obstacle region by disjoint discs is that it may prevent the robot from accessing to a large subset of the free space. This is in particular the case if the C-obstacle region is made of elongated objects. An elliptical repulsive potential, as defined below, is a compromise between the repulsive potential of Subsection 1.3, which tends to create large local minima, and the circular repulsive potential presented above, which prevents access to a large subset of the free space.

Consider again the simple case where the C-obstacle region $\mathcal{CB}$ is a rectangle. Let $L$ and $H$ denote the lengths of the long and small sides of $\mathcal{CB}$. We select the coordinate frame in $\mathcal{C} = \mathbf{R}^2$ with its origin at the center of $\mathcal{CB}$ and its $x$-axis pointing along the long side of $\mathcal{CB}$. The elliptical repulsive potential generated by $\mathcal{CB}$ is defined in two steps [Volpe and Khosla, 1987]. First, the equipotential contours are specified. Second, the value of the potential on each equipotential contour is defined.

An $n$-ellipse ($n \geq 1$) is a curve with the following equation:

$$\left(\frac{x}{a}\right)^{2n} + \left(\frac{y}{b}\right)^{2n} = 1 \qquad (2)$$

where $a$ is the semi-major axis and $b$ is the semi-minor axis. If we set:

$$a = \frac{L}{2}\left(2^{\frac{1}{2n}}\right) \quad \text{and} \quad b = \frac{H}{2}\left(2^{\frac{1}{2n}}\right) \qquad (3)$$

we get an $n$-ellipse that touches $\mathcal{CB}$ at its four corners. In addition, the surface of the area between the rectangle and the $n$-ellipse is minimal (over all the $n$-ellipses passing through the four corners of $\mathcal{CB}$). When $n \to \infty$, the $n$-ellipse converges toward the boundary of $\mathcal{CB}$.

In order to make the $n$-ellipse become a circle when $n = 1$, we multiply

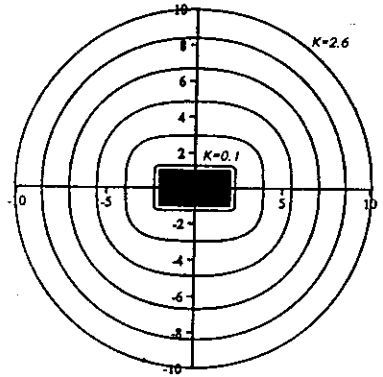



**Figure 11.** This figure shows elliptical equipotential contours for values of $K$ between 0.1 and 2.6, and $\alpha = 1.5$. The contours are close to the rectangle when $K$ is small and almost circular when $K$ is large [Volpe and Khosla, 1987].

the term in $y$ by $(b/a)^2$. The $n$-ellipse becomes:

$$\left(\frac{x}{a}\right)^{2n} + \left(\frac{b}{a}\right)^{2}\left(\frac{y}{b}\right)^{2n} = 1 \quad (n \geq 1).$$

This $n$-ellipse is a circle if $n = 1$. For every $n \geq 1$, it contains $\mathcal{CB}$, and it converges toward the boundary of $\mathcal{CB}$ when $n \to \infty$.

For a given $n$, we treat the quantity:

$$\xi_n(x,y) = \left[\left(\frac{x}{a}\right)^{2n} + \left(\frac{b}{a}\right)^{2}\left(\frac{y}{b}\right)^{2n}\right]^{\frac{1}{2n}} - 1$$

as a pseudo-distance between the configuration $(x, y)$ and $\mathcal{CB}$. It is zero in the $n$-ellipse curve and grows to infinity for configurations moving away from this curve. For every given constant value $K$, the curve $\xi_n(x, y) = K$ is taken as a repulsive equipotential, for some $n$ determined by:

$$n = \frac{1}{1 - e^{-\alpha K}}$$

where $\alpha$ is an adjustable parameter. Thus, when $K \to \infty$, $n \to 1$, and the equipotential tends toward a circle. When $K \to 0$, $n \to \infty$, and the equipotential tends toward the boundary of $\mathcal{CB}$. The parameter $\alpha$ determines how quickly the "$n$-ness" of the ellipse increases when $K \to 0$.



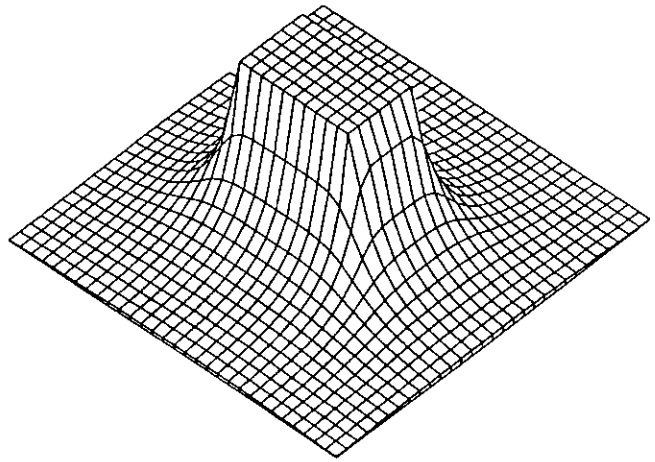


**Figure 12.** This figure shows the elliptical repulsive potential generated by a rectangular C-obstacle with $\alpha = \beta = 1$ and $\eta = 1$.

Figure 11 shows equipotential contours around $\mathcal{CB}$ for various values of $K$. Let $R(K)$ be the repulsive equipotential contour[11] corresponding to the pseudo-distance $K$.

Now that the form of the repulsive equipotential contours has been specified, we define the value of the repulsive potential in each of them. It is desirable that the repulsive potential of $\mathcal{CB}$ have a limited range of influence, with an inverse dependence on the distance to the C-obstacle. This can (almost) be achieved by taking:

$$\forall (x,y) \in R(K): \quad \mathbf{U}_{rep}(x,y) = \eta \frac{e^{-\beta K}}{K}$$

where $\eta$ is a scaling factor. The exponential term makes the potential drop toward zero faster than $K^{-1}$, when $K$ increases. Figure 12 shows the repulsive potential $\mathbf{U}_{rep}$ generated by the rectangular C-obstacle $\mathcal{CB}$.

When the above repulsive potential is added to the attractive potential (with $\mathbf{q}_{goal}$ located as in Figure 10), a local minima may still be created

[11] The use of $n$-ellipsoids in the definition of repulsive potentials was first proposed in [Khatib, 1986]. Khatib suggested that a rectangular C-obstacle be approximated by an $n$-ellipsoid, for some fixed $n$. In contrast, here $n$ varies from 1 to infinity.

"uphill" from $\mathcal{CB}$. However, the size of the domain of attraction of this minimum can be tuned by setting $\eta$, $\alpha$ and $\beta$ appropriately. It is possible to avoid the local minimum by making the minimum of the total potential along the $y$-axis happen at a point $(0, y_0)$ where the radius of curvature of the repulsive equipotential contour is smaller than or equal to the radius of the attractive equipotential contour. (Then, in the plane, $(0, y_0)$ is a saddle point.) Since this happens with a non-circular repulsive equipotential contour, less free space is wasted than with circular repulsive potentials. The wasted space is minimum when the parameters are set in so that $y_0$ is as close as possible to $\mathcal{CB}$. The wasted free space may still be too large (e.g. it may include the goal configuration), and the parameters may be set differently. Then, the domain of attraction of the created local minimum is typically smaller than for the local minimum created by the repulsive potential function of Subsection 1.3.

Nevertheless, elliptical potentials have several drawbacks which make their application to practical problems difficult. First, they can only be generalized to simple convex shapes (e.g. trapezoid) in two- and three-dimensional spaces, using superquadric formulation with non-constant scaling parameters (see [Khosla and Volpe, 1988]). Second, they require C-obstacles to be sufficiently distant from each other, so that no C-obstacle is within the distance of influence of another C-obstacle. Third, their definition does not provide a direct way to compute the potential at a given configuration; it requires interpolating among precomputed values at nearby configurations.

One can also compute the elliptical repulsive potential in the workspace, apply it at selected control points in the robot, and thus construct a configuration space potential (see Subsection 2.3).

# 5 Randomized Planning

In Section 3 we have described simple potential-field-based planning techniques. These techniques, however, have a limited ability to deal with local minima of the potential function. On the other hand, in Section 4 we have seen that the construction of a navigation function, i.e. a potential field with no local minimum other than the goal configuration, is a difficult problem that has a known solution only when the C-obstacles have simple shapes and/or when the dimension of the configuration space is small ($m = 2$ or 3). In this section we present a powerful planner, which we call RPP (for Randomized Path Planner), that combines some of the techniques described above with randomized search techniques. This planner has been implemented and successfully experimented with difficult planning problems [Barraquand and Latombe, 1989a and 1990a].



This section serves three purposes. First, it introduces a new technique — randomized search — within the framework of the potential field approach. Second, it presents an integrated view of what a planner based on the potential field approach looks like. Indeed, so far we have only described separate components of such a planner. Third, RPP will also be applicable to problems involving high-dimensional configuration spaces studied in the next chapter (multiple robots, articulated robots).

## 5.1 Overview

The techniques presented below do not require any specific potential function to be used. However, in order to make our presentation more precise, we assume that this function U is a grid potential computed as described in Subsection 4.2.3 and we use the same notations. We denote the coordinate axes in the grid $\mathcal{GC}$ by $x_1$, ..., $x_m$, and the grid increment along any $x_i$-axis by $\Delta_i$. Since the dimension $m$ of the grid may be large, the planner has no explicit representation of $\mathcal{GC}$, e.g. in the form of an array. We assume that $\mathbf{U} \geq 0$ and that any configuration $\mathbf{q}$ such that $\mathbf{U}(\mathbf{q}) = 0$ is a goal configuration.

In a first approximation, we may imagine that RPP constructs and searches a graph $G$ whose nodes are local minima of the potential function U. Two minima are connected by a link in the graph if the planner has constructed a path joining them.

The planner starts at the input initial configuration $\mathbf{q}_{init}$. From there, it executes a **best-first motion**, i.e. it follows the steepest descent of the potential U. The best-first motion stops when it reaches a minimum $\mathbf{q}_{loc}$. If $\mathbf{U}(\mathbf{q}_{loc}) = 0$, the problem is solved and the algorithm returns the constructed path. Otherwise, it attempts to escape from the local minimum by executing a series of **random motions** issued from $\mathbf{q}_{loc}$. Each random motion is immediately followed by a new best-first motion that attains a minimum. If this minimum is different from $\mathbf{q}_{loc}$, it is