Robotica (2000) volume 18, pp. 123–142. Printed in the United Kingdom © 2000 Cambridge University Press

Two novel approaches for unmanned underwater vehicle path planning: constrained optimisation and semi-infinite constrained optimisation

Yongji Wang,* David M. Lane† and Gavin J. Falconer†

(Received in Final Form: July 31,1999)

SUMMARY

In this paper, two novel approaches to unmanned underwater vehicle path planning are presented. The main idea of the first approach, referred to as *Constrained Optimisation (CO)* is to represent the free space of the workspace as a set of inequality constraints using vehicle configuration variables. The second approach converts robot path planning into a *Semi-infinite Constrained Optimisation (SCO)* problem. The function interpolation technique is adopted to satisfy the start and goal configuration requirements. Mathematical foundations for Constructive Solid Geometry (CSG), Boolean operations and approximation techniques are also presented to reduce the number of constraints, and to avoid local minima. The advantages of these approaches are that the mature techniques developed in optimisation theory which guarantee convergence, efficiency and numerical robustness can be directly applied to the robot path planning problem. Simulation results have been presented.

KEYWORDS: Robotics, Autonomous underwater vehicle; path planning; obstacle avoidance; Non-linear programming; Semi-infinite constrained optimisation.

1. INTRODUCTION

Unmanned Underwater Vehicles are extensively used for the inspection, maintenance and repair of offshore structures in both deep and shallow water.[1-5] Current generation work-class vehicles are generally equipped with one or two manipulators with between 5 and 9 degrees of freedom (DOF), and are exclusively tele-operated from the surface (Figure 1). As the complexity, volume and depth of work expands, the use of increased automation to cut costs has become a research focus. Unlike manufacturing, the underwater domain is an unstructured environment, where reactive behaviour to unexpected events is required. This requires relevant sensing, planning and control methods to provide the vehicle with reasonable behaviours, and to avoid frequent operator intervention.

This paper considers the problem of reactively planning path for the vehicle through a cluttered environment, such as around a subsea structure. Although seemingly trivial, it has proved notoriously difficult to find techniques which work reliably in the presence of multiple obstacles. The problem is well known as a find-path problem which can be described as follows: Given a subsea vehicle with an initial configuration, a goal configuration, and a set of obstacles located in the workspace, find a continuous, collision-free path from the initial configuration to the goal configuration for the vehicle.

A significant and varied effort has been made on this seemingly simple, but in fact very complicated find-path problem.[6-25] Various methods for dealing with the basic find-path problem and its extensions, such as Vgraph,[14] Voronoi diagram,[12] exact cell decomposition,[6] approximate cell decomposition,[12] potential field approach,[10] genetic-based approach,[7] and optimisation-based approach[16] have been developed. A systematic discussion on these methods can be found in references.[5,12,21]

The potential field approach was pioneered by Khatib[11] and has been followed by many researchers.[10,12,13] It was thought more efficient than other traditional approaches. It treats the robot as a point in configuration space under the influence of an artificial field *U,* constructed around the obstacles which present potential collision hazards in the workspace. The potential function is typically defined as the sum of an attractive potential pulling the robot toward the goal configuration and a repulsive potential pushing the robot away from the obstacles. Path planning is performed iteratively. At each iteration, the direction of the artificial force induced by the potential function at the current configuration is regarded as the most promising direction of motion, and the path generation proceeds along this direction by some increment.

Potential field approach was originally developed as an on-line collision avoidance technique when the robot does not have a prior model of the obstacles, but senses them during motion execution. Although efficient, it has some disadvantages. First this approach is not a purely geometric one. The potential field resulting from both the obstacles and the goal point are artificial, depending on the choices of the weighted parameters in the definitions. This means that even for the same robot and environment, different definitions of the potential field may lead to contrasting results. Second, since an on-line potential field approach essentially acts as a fastest descent optimisation procedure, it may get stuck at a local minimum of the potential field rather than reach the goal configuration. Dealing with local minima is

* Centre for Systems and Control, Department of Mechanical Engineering, Glasgow University, Glasgow G12 8QQ, Scotland, (UK)
†Ocean Systems Laboratory, Department of Computing & Electrical Engineering, Heriot-Watt University, Edinburgh, EH14 5AS, Scotland, (UK)
E-mail: ywang@mech.gla.ac.uk, dml@cee.hw.ac.uk, gjf@cee.hw.ac.uk

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

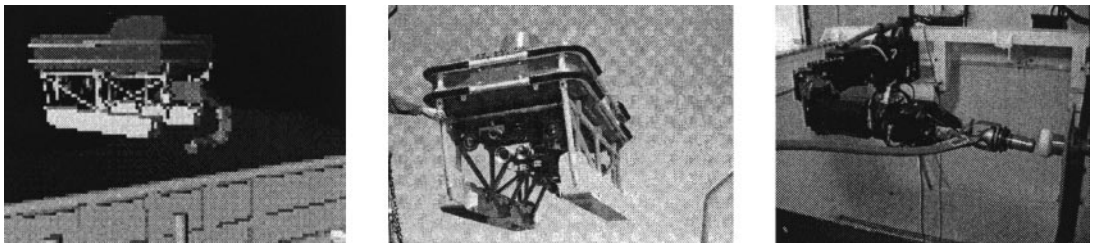


Fig. 1. ANGUS 003, an unmanned underwater vehicle, and the Slingsby TA9, 7 degree of freedom manipulator.

the major issue one has to face. There has not been a sound theoretical proof that the goal configuration is at least one of the minimum points. It seems to be difficult to define a versatile potential function which guarantees that the goal point is the global minimum of the total potential field.

In this paper, we present two novel robot path planning approaches based on optimisation theory which can overcome the shortcomings of the potential field approaches but maintain their efficiency. These approaches do not need the definition of the potential fields for both the obstacles and the desired goal configuration. We use Constructive Solid Geometry (CSG) to represent the obstacles as a set of inequalities. The ideas of the first approach, referred here to as the *Constrained Optimisation* (*CO*) approach, include: 1. the free space of the robot workspace is represented as inequality constraints and 2. the goal configuration is designed as the unique global minimum point of an objective function. Thus the problem is transferred into a constrained optimisation problem. The initial configuration is treated as the start point for optimisation problem. Then a numerical algorithm developed for solving nonlinear programming problems can be applied to solve the robot motion planning problem. Every immediate point generated using the nonlinear optimisation search method is guaranteed to be in free space and therefore is collision free. The second approach converts the robot path planning problem into a *Semi-infinite Constrained Optimisation* (*SCO*) problem. This is realised by the use of function interpolation technique satisfying the start and goal configuration requirements.

In any robot path planning method, obstacle representation is the first thing to be considered. For more detailed description of 3D object representation, see the references.[26-31] In this paper obstacles are represented as implicit functions (inequalities) which lead to the use of efficient search algorithms. The context of this paper covers a wide range of subjects such as CAD, Computer Graphics and nonlinear programming theory. We describe only the necessary fundamentals therefore, relying on the references for more detailed descriptions. Our focus will be on the interconnection of these theoretical areas.

This paper is organised as follows: Section 2 gives a brief description of the formulations for both the constrained optimisation and semi-infinite constrained optimisation. In Section 3, the principle of the potential field approach is presented and the reason for local minima is discussed. The Constructive Solid Geometry approach for representing obstacles as inequalities is presented in Section 4. In Section 5, we show how to convert the robot path planning problem into the CO problem and simulation results are presented. Similarly we show how to convert the robot path planning problem into the SCO problem in Section 6, and simulation results are also given. Finally conclusions are presented in Section 7.

2. TWO DIFFERENT FORMULATIONS OF OPTIMISATION PROBLEMS

Optimisation concerns the minimisation or maximisation of a function subject to different types of constraints (equality or inequality). The function is referred to as the objective function. Optimisation has a very sound theoretical background and many applications; finding the roots of a set of nonlinear equations, detecting the intersections of solid objects, curve fitting and so on.

The main difficulty in solving nonlinear programming problems is the need to develop a convergent, efficient, and robust algorithm. This is also one of the problems faced by the potential field approach. Fortunately, the development of a systematic theory on optimisation with non-linear constraints has been an active field during the past thirty years, and many efficient descent search approaches suitable for various problems have been developed. Optimisation techniques have now become a mature subject. In the following, we give a brief introduction to optimisation problems because the emphasis of this paper will be focused on how to convert the path planning problem into a standard nonlinear programming problem, rather than on the details of the nonlinear programming theory which are easily found.[32-37]

Most algorithms designed to solve large optimisation problems are iterative in nature. Typically, in seeking a vector that solves the optimisation problem, an initial vector $\boldsymbol{x}_0$ is selected and the algorithm generates an improved vector $\boldsymbol{x}_1$, which means that $f(\boldsymbol{x}_0) \leq f(\boldsymbol{x}_1)$, where $f(\boldsymbol{x})$ represents the objective function. The process is repeated and a still better solution $\boldsymbol{x}_2$ is found. Continuing in this fashion, a sequence of ever-improving points $\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k, \ldots$, is found that approaches a solution point $\boldsymbol{x}^*$, referred to as the minimum (maximum) point.

There are mainly four different types of optimisation problem: *Linear Programming, Unconstrained Problems, Constrained Problems* and *Semi-infinite Constrained Problems.* The last three parts together comprise the subject of *Nonlinear Programming.* When applying optimisation theory to a practical problem, the most important requirement is to formulate the problem as one of the standard forms: then the algorithms specially developed for the corresponding type of problem can be applied.

MATLAB is a widely used software package for solving problems encountered in control, signal processing, optimi-

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

sation, and other scientific and engineering numerical calculations.[37] Many standard optimisation algorithms have been implemented in the MATLAB *optimisation* toolbox which can be used to eliminate the need for designers to create and re-create commonly used programmes – a big labour saving. There are many different optimisations, each suitable for solving different problems. For completeness, Tables I and II give the different problem formulations provided by the MATLAB optimisation toolbox. Among them, the most general two are the constrained optimisation (CO) and the semi-infinite constrained optimisation (SCO), both of which will be used for robot path planning. Below is a brief description of the CO and SCO formulations.

2.1. Constrained optimisation (CO)

The general constrained optimisation problem can be described as follows:

Find an optimal point $\boldsymbol{x}^*$ which minimises the function $f(\boldsymbol{x})$, with $\boldsymbol{x} \in \Re^n$, subject to:

$$\begin{aligned} & g_i(\boldsymbol{x}) = 0,\ i = 1, 2, \ldots\ldots, m \\ & g_j(\boldsymbol{x}) \le 0,\ j = 1, 2, \ldots\ldots, n \\ & \boldsymbol{x}_l \le \boldsymbol{x} \le \boldsymbol{x}_u \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is an n-dimensional vector of unknowns, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, and f, g_i, $i = 1, 2, \ldots, m$ and g_j, $j = 1, 2, \ldots, n$, are real-valued functions of the variables $(x_1, x_2, \ldots, x_n)$. $\boldsymbol{x}_l$ and $\boldsymbol{x}_u$ are the lower and upper bounds of $\boldsymbol{x}$, respectively. The function f is the objective of the problem, and the equalities and inequalities are constraints.

2.2. Semi-infinite constrained optimisation (SCO)

The semi-infinite constrained optimisation problem is to find the minimum of a semi-infinitely constrained scalar function of several variables $\boldsymbol{x}$ starting at an initial estimate $\boldsymbol{x}_s$. This problem is mathematically stated as:

Minimise $f(\boldsymbol{x})$, $\boldsymbol{x} \in \Re^n$, subject to:

$$\begin{aligned} & \boldsymbol{g}(\boldsymbol{x}) = 0 \\ & \boldsymbol{\Phi}_j(\boldsymbol{x}, \boldsymbol{w}_j) \le 0,\ j = 1, 2, \ldots, n \\ & \boldsymbol{x}_l \le \boldsymbol{x} \le \boldsymbol{x}_u \\ & \text{for all } \boldsymbol{w}_j \in \Re^2 \end{aligned} \tag{2}$$

where $\boldsymbol{\Phi}_j(\boldsymbol{x}, \boldsymbol{w}_j)$ is a continuous function of both $\boldsymbol{x}$ and an additional set of variables, $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$. The variables $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$ are vectors of at most length two. The aim is to minimise $f(\boldsymbol{x})$ so that the constraints hold for all possible values of $\boldsymbol{\Phi}_j(\boldsymbol{x}, \boldsymbol{w}_j)$. Since it is impossible to calculate all possible values of $\boldsymbol{\Phi}_j(\boldsymbol{x}, \boldsymbol{w}_j)$, a region must be chosen for $\boldsymbol{w}_j$ over which to calulate an appropriately sampled set of values. $\boldsymbol{x}$ is referred to as the unknown variable and $\boldsymbol{w}_j$ as the independent variable.

The difference between the two formulations is obvious. The latter allows an extra set of independent variables $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$ to be included in the constraints rather than only the unknown variable $\boldsymbol{x}$, as is the case in the former. Note that the former is a special case of the latter, and the latter cannot be converted into the former. For more details (see reference 37).

The procedure for solving such a semi-infinite optimisation problem with nonlinear constraints is as follows:

(a) assign an initial guess point $\boldsymbol{x}_s$ and a region for $\boldsymbol{w}_j$
(b) use a search algorithm to find the optimum solution $\boldsymbol{x}^*$ and the corresponding minimum objective function $f(\boldsymbol{x}^*)$.

3. POTENTIAL FIELD APPROACH

Potential field approach consists of defining a potential field and then finding a search algorithm in order to reach the goal configuration. The main issue for this approach is local

Table I: Nonlinear minimisation problems*

Type	Notation
Unconstrained scalar	min $f(\boldsymbol{x})$
Unconstrained	min $f(\boldsymbol{x})$
Constrained	min $f(\boldsymbol{x})$ such that $\boldsymbol{g}(\boldsymbol{x}) \le 0$
Goal	min γ such that $\boldsymbol{f}(\boldsymbol{x}) \le \boldsymbol{Goal}$
Minmax	min{max $\boldsymbol{f}(\boldsymbol{x})$} such that $\boldsymbol{g}(\boldsymbol{x}) \le 0$
Nonlinear least squares	min $\sum \{\boldsymbol{f}(\boldsymbol{x}).^* \boldsymbol{f}(\boldsymbol{x})\}$
Nonlinear equations	$\boldsymbol{f}(\boldsymbol{x}) = 0$
Semi-finite constrained	min $f(\boldsymbol{x})$ such that $\boldsymbol{g}(\boldsymbol{x}) \le 0$ & $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{w}) \le 0\ \forall\ \boldsymbol{w}$

Table II: Matrix problems*

Type	Notation
Non-neg. least squares	min $\|\boldsymbol{Ax}\text{-}\boldsymbol{b}\|^2$ such that $\boldsymbol{x} \le 0$
Quadratic program	min$\{0.5\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}^T\text{-}\boldsymbol{c}^T\boldsymbol{x}\}$ such that $\boldsymbol{Ax}\text{-}\boldsymbol{b} \le 0$
Linear program	min $\boldsymbol{c}^T\boldsymbol{x}$ such that $\boldsymbol{Ax}\text{-}\boldsymbol{b} \le 0$

* The routines are designed to work with sclars, vectors, and matrices. Matrices are indicated by upper-case bold letters, vectors by lower-case bold letters, and scalars by plain lower-case letters. The MATLAB notion. * indicates element-wise multiplications.

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

minima. In this section, we examine the causes of local minima and explain the difficulties encountered in avoiding them. Some typical simulation results are also presented.

3.1. Principle of potential field approach

The potential field is generated in two stages. The first is the definition of an attractive field which arises from the goal point and functions as drawing the robot towards that point. The next stage is the definition of a repulsive field which arises from the obstacles and functions as pushing the robot away from those obstacles. In the following we will show that potential field approach is actually an unconstrained opimisation problem. We shall furthermore examine the factors which may cause local minima.

The purpose of the attractive potential field, $f_{att}(\boldsymbol{q})$, is to cause the robot to move towards the goal point, and therefore it should have only one minimum at this point. One simple solution is to use a parabolic well:[12]

$$f_{att}(\boldsymbol{q}) = 0.5\,\eta\rho^2_{goal}(\boldsymbol{q}) \tag{3}$$

where $f_{att}(\boldsymbol{q})$ is the value of the field at configuration $\boldsymbol{q}$, η is a positive scaling factor and $\rho_{goal}(\boldsymbol{q}) = ((\boldsymbol{q}-\boldsymbol{q}_g)(\boldsymbol{q}-\boldsymbol{q}_g)^{\mathrm{T}})^{0.5}$ is the Euclidean distance between the configuration $\boldsymbol{q}$ and the goal point $\boldsymbol{q}_g$.

The definition of the attractive potential field is not unique. An alternative definition is the conical well:

$$f_{att}(\boldsymbol{q}) = 0.5\,\eta\rho_{goal}(\boldsymbol{q}) \tag{4}$$

Figures 2(a) and (b) show a typical example of the attractive field $f_{att}(\boldsymbol{q})$ resulting from (3) and (4) respectively, when the configuration of the robot is represented as a point moving on a plane, i.e. $\boldsymbol{q} = (x, y)$.

The most-widely used definition for the repulsive potential is as follows:[12]

$$f_{rep}(\boldsymbol{q}) = \begin{cases} 0.5\,\mu\left(\dfrac{1}{\rho(q)} - \dfrac{1}{\rho_0}\right) & \text{if} \quad \rho(q) \le \rho_0 \\ 0 & \text{if} \quad \rho(q) \ge \rho_0 \end{cases} \tag{5}$$

where μ is a positive scaling factor, $\rho(\boldsymbol{q})$ is the smallest distance between the configuration $\boldsymbol{q}$ and any obstacle, and ρ_0 is a positive constant. ρ_0 is known as the *distance of influence* of an obstacle, and defines the range over which an obstacle will exert a repulsive force. Figures 3(a) and (b) show the repulsive potential set up for an obstacle with both a large and a small *distance of influence.*

Once the attractive and the repulsive potentials are defined, the total potential field $f_{tot}(\boldsymbol{q})$ is found simply by summing $f_{att}(\boldsymbol{q})$ and $f_{rep}(\boldsymbol{q})$.

$$f_{tot}(\boldsymbol{q}) = f_{att}(\boldsymbol{q}) + f_{rep}(\boldsymbol{q}) \tag{6}$$

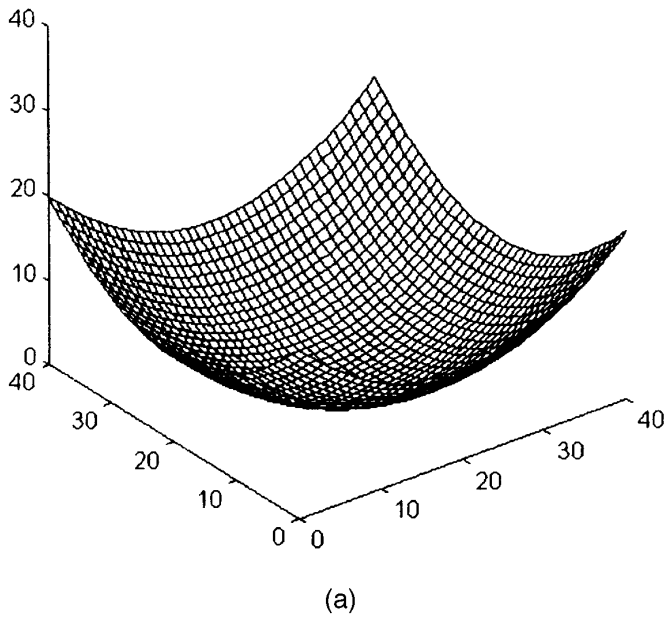
The aim of the potential field approach is to find the minimum point of the total potential function (6) which, if there are no other local minima, will be the goal point. Treating the start point $\boldsymbol{q}_{start}$ as $\boldsymbol{q}_0$ an iteration of the following form must be designed:

$$\boldsymbol{q}_{i+1} = \boldsymbol{q}_i + \alpha_i \boldsymbol{S}_i \tag{7}$$

where α_i and $\boldsymbol{s}_i$ are determined as a correction to $\boldsymbol{q}_i$ such that $f_{tot}(\boldsymbol{q}_{i+1}) \le f_{tot}(\boldsymbol{q}_i)$. One may imagine $\boldsymbol{q}_i$ as a point in n dimensional space and $\boldsymbol{s}_i$ as a *direction of search* emanating from $\boldsymbol{q}_i$ with $\boldsymbol{q}_{i+1}$ as the optimum point along this direction, as shown in Figure 4. α_i is usually chosen to minimise $f_{tot}(\boldsymbol{q}_i + \alpha_i\, \boldsymbol{s}_i)$. There are many choices for $\boldsymbol{s}_i$. One of the oldest methods is that of *Steepest Descent* in which $\boldsymbol{s}_i$ is chosen simply as the downhill gradient vector $\boldsymbol{s}_i = -[\partial f_{tot}/\partial q_1, \partial f_{tot}/\partial q_2, \ldots, \partial f_{tot}/\partial q_n]$. It is based on the fact that, local to the current approximation, the direction $\boldsymbol{s}_i$ is that along which $f_{tot}(\boldsymbol{q}_i + \alpha_i \boldsymbol{s}_i)$ decreases most rapidly. Therefore the potential field approach is actually formulating the robot path planning problem as an unconstrained optimisation problem.

3.2. Causes for local minima

From the above discussions we can see that, for the potential field approach, the local minima are those of $f_{tot}(\boldsymbol{q})$. They arise from the definition of the potential field function. In



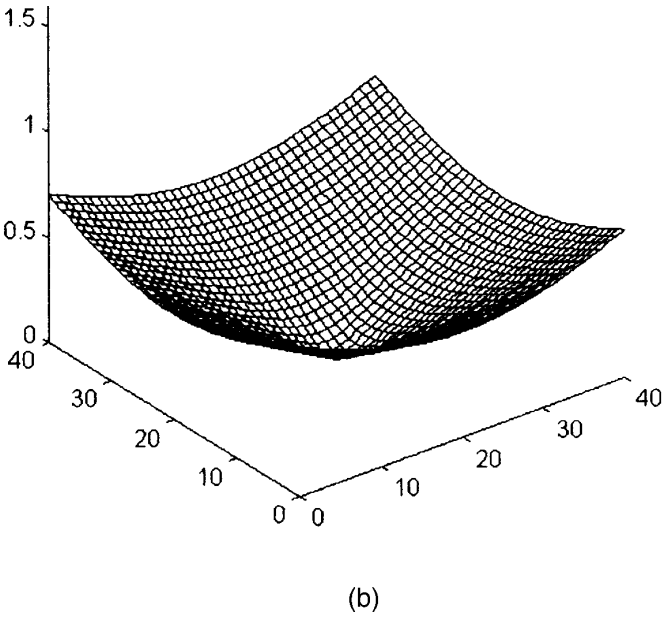


Fig. 2. (a) Parabolic potential well, (b) Conical potential well.

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

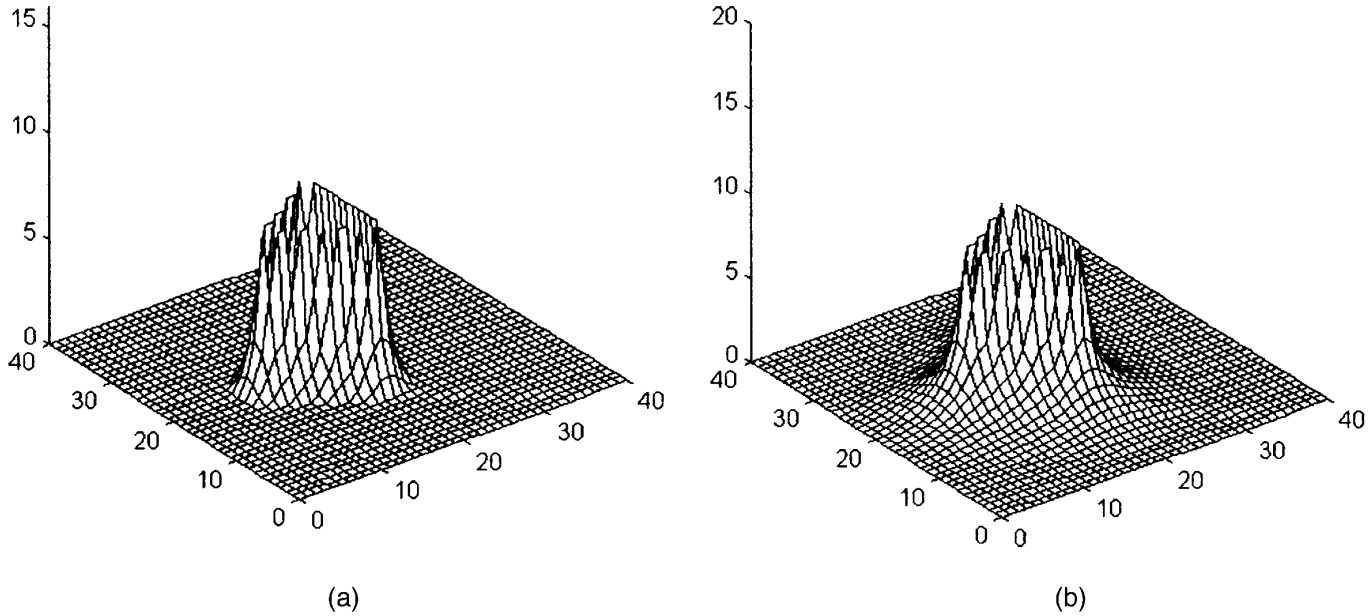


Fig. 3. (a) Repulsive potential with $\rho_0 = 4$, (b) Repulsive potential with $\rho_0 = 12$.

order to have no local minima, the goal point must be designed either as a unique minimum point which is usually called the *global minimum point* in optimisation theory, or as a *local minimum point* within some neighbourhood of the start point (there is a mathematically precise definition of local minimum in optimisation). Figures 5, 6, 7 and 8 show some typical simulation results using the potential field approach with the following parameters:

- Parabolic attractive potential well
- Attractive potential scaling factor, $\eta = 0.025$
- Repulsive potential scaling factor, $\mu = 100$
- Obstacles' distance of influence, $\rho_0 = 8$.

In each figure, (a) gives the total potential and (b) presents the path generated using steepest descent search. These results indicate that potential field methods do work sometimes, as shown in Figure 5. The obstacle is successfully avoided and no local minima are present. However, when the start and the goal points are in the two opposite sides of the obstacle, the potential field approach fails, as shown in Figure 6. Even when both the start and the goal points are located at the same side of the obstacle and the goal point is close to the obstacle, a path can not be found due to the inappropriate choice of the relative values of η and μ (see Figure 7). In Figure 8 the two obstacles are close enough together for their repulsive fields to combine and they act like a single concave obstacle. As a result, a local minimum is found and the goal is not reached. When the number of obstacles increases, the local minima problem becomes more serious.

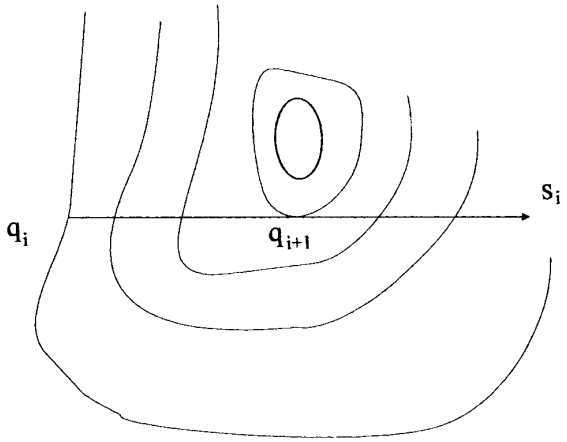


Fig. 4. The search along a line for a minimum.

Many factors affect the distribution of the local minima: the goal position, the position and shape of the obstacles and the definition of the function type of both the attractive and the repulsive potentials as well as the choice of values for parameters η, μ and ρ_0. The arbitrariness of the number of obstacles in the workspace, the distribution of these obstacles and the positions of the start and goal points means that it is difficult to design a robust potential field which guarantees that the goal point is always a unique global minimum point of the total potential field or at least a local minimum within some neighbourhood of the start point. Our experience has shown that even for the same robot (start and goal) and environment (shape and position of the obstacles), different definitions of the potential field may lead to contrasting results. For example, one definition may give a successful path and the other may report failure. On the other hand, one definition of the potential field may work for one kind of robot and environment, but fail for another.

4. OBJECT REPRESENTATION USING CONSTRUCTIVE SOLID GEOMETRY (CSG)

Clearly the first thing for robot path planning is to give each of the objects in the workspace a mathematical representation. Modelling and manipulation of objects is the research task of Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), and Computer Graphics (CG). A solid model should contain an informationally complete description of the geometry and topology of a 3-D object.[28] A successful modelling system, in addition to many other

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

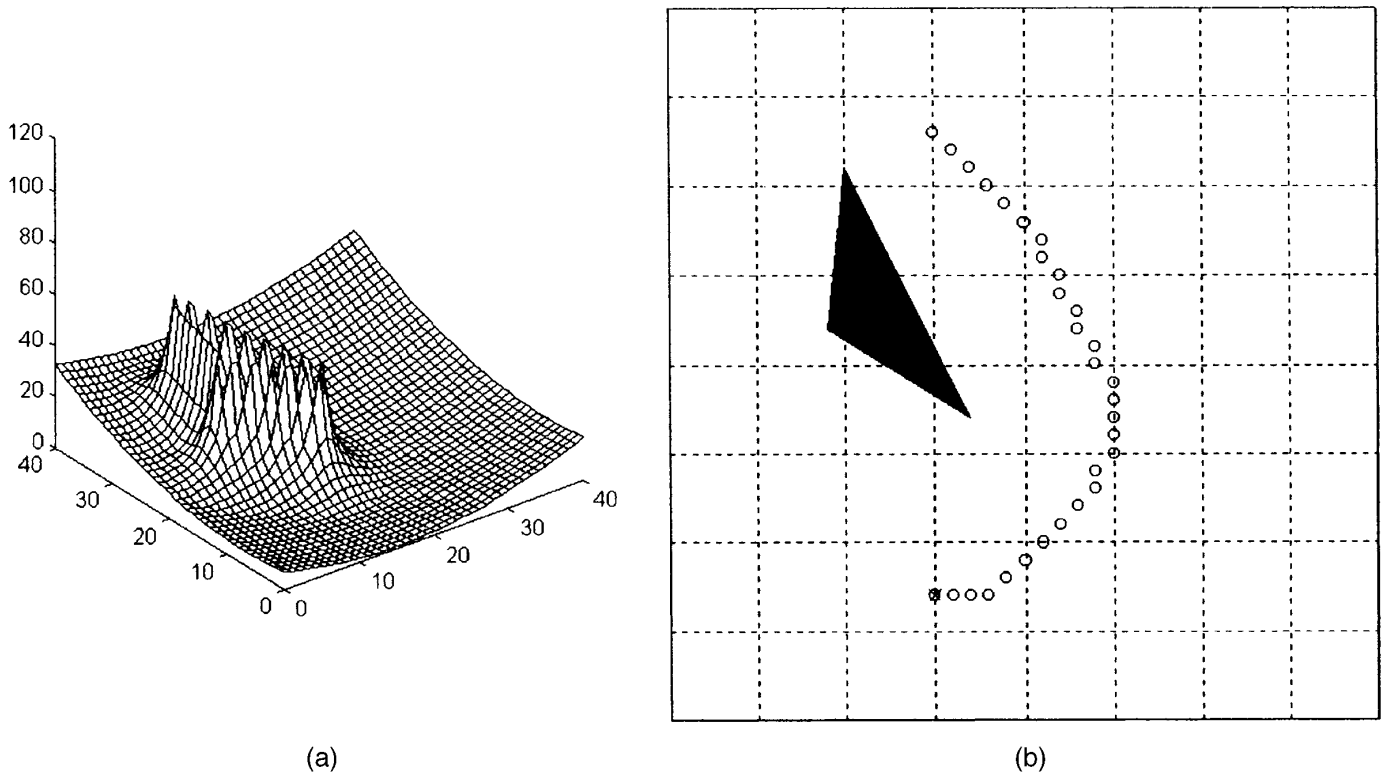


Fig. 5. (a) Potential function (b) Path generated by using steepest descent search.

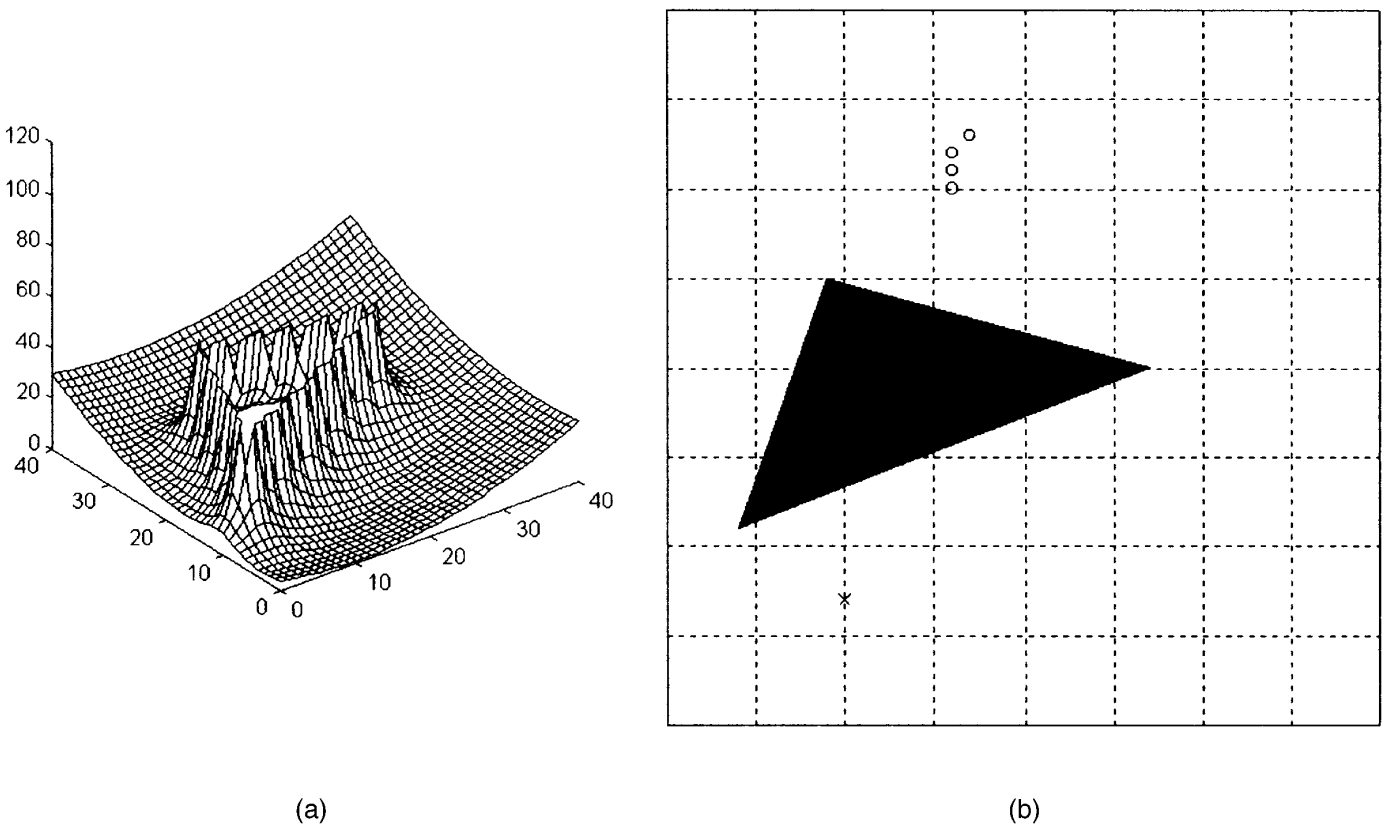


Fig. 6. (a) Potential function. (b) Path generated by using steepest descent search.

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

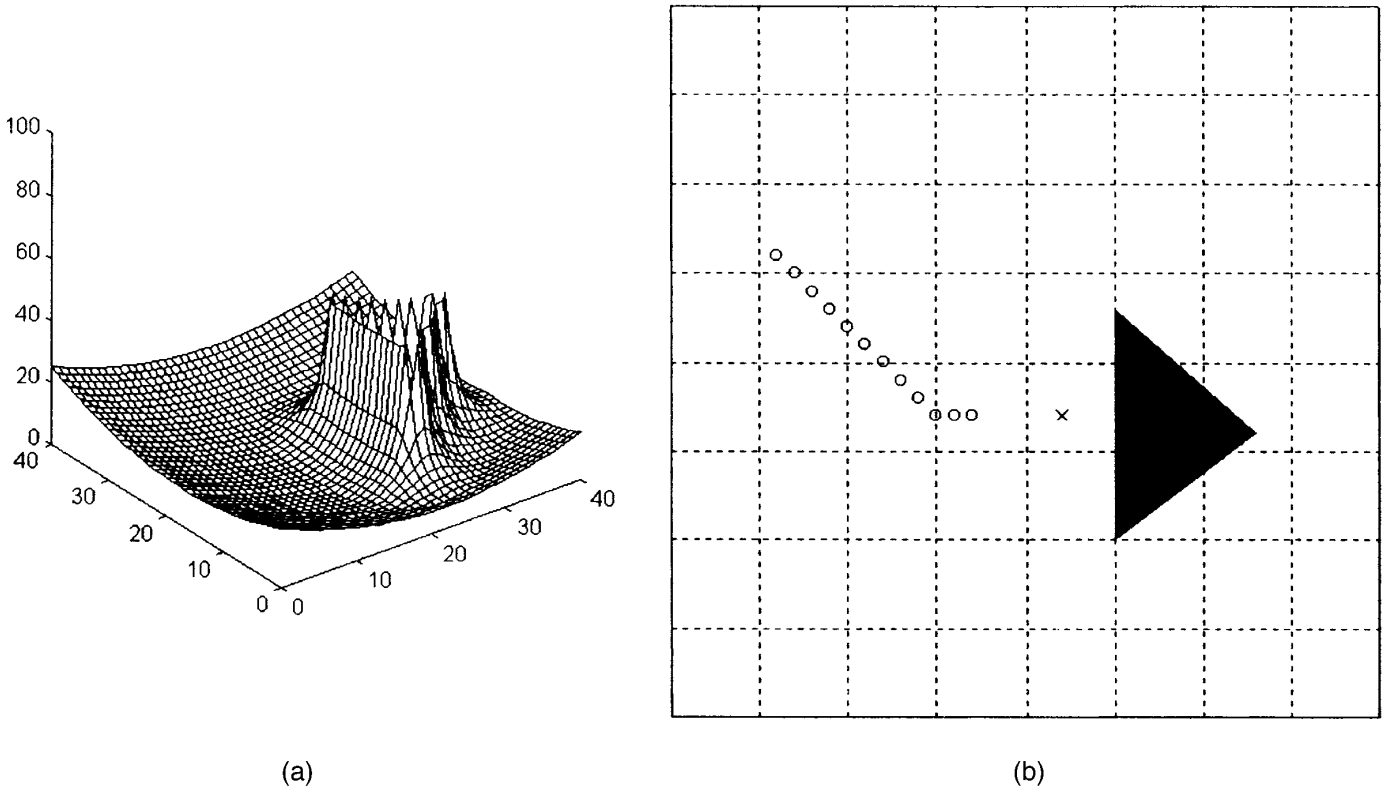


Fig. 7. (a) Potential function. (b) Path generated by using steepest descent search.

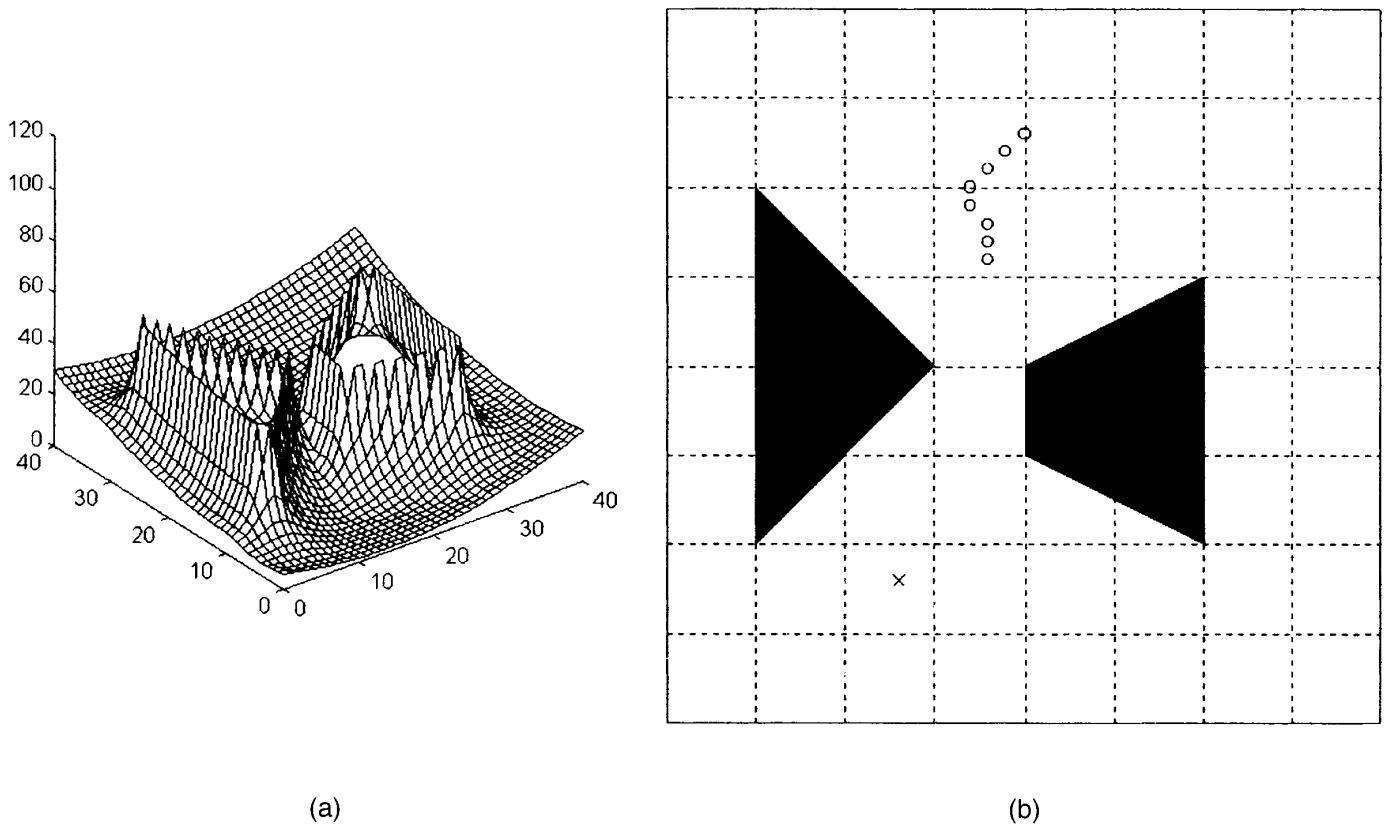


Fig. 8. (a) Potential function. (b) Path generated by using steepest descent search.

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

features, must be capable of representing the object's surface and be able to unambiguously determine whether a point is on the "inside" or "outside" of the object. In CAD, CAM and CG, there are three traditional categories of solid modelling systems, namely boundary representation (B-rep), spatial decomposition, and constructive solid geometry (CSG). Different modelling approaches have their own advantages and disadvantages. Chiyokura[28] gives detailed descriptions, comparisons, advantages, and disadvantages of these and other modelling approaches.

For the robot motion planning problem using nonlinear programming theory, CSG together with the approximation approach discussed later are used to represent the free space as inequality constraints. The CSG approach seems to be more natural and promising because most of the practical objects can be described by very simple geometric surfaces such as the half space, sphere and cylinder which are all CSG primitives. In addition, Boolean operations are much easier to apply to these primitives.[27,31] The CSG approach does not need a large amount of data to define surface points as well as the topological relationships among surfaces, edges and points. In CSG objects are built by moving, joining, and cutting primitive objects. More specifically, primitive solids such as half-spaces, spheres, cylinders, cuboids, and superellipsoids are combined using Boolean operations which include union, intersection, and difference. The resulting object has a single, smooth, and closed expression for its surface. This approach is good, therefore, for modelling typical subsea structures (manifolds, platforms, pipelines, wellheads, benthic stations). It is, moreover, a convenient representation used by CAD/CAM modelling systems for structure design. Hence data for typical environments could be readily available in an appropriate form.

In the following, mathematical foundations of CSG and the Boolean operations are presented. Particularly, attention is focused on the problems encountered in the implementation of the approximation of the Boolean operations for slabs. The definition of a defining function is first introduced and then the Boolean operations using the defining function are presented. An approximate method for the Boolean operations (intersection and union) and some examples are also given. The material presented in this section is mainly based on.[27,28,31]

4.1. Definition of the defining function

For a solid $\boldsymbol{S}$ considered in the 3D Euclidean space $\boldsymbol{E}^3$, the set of its interior points is denoted by $\boldsymbol{I}$, the set of its boundary points by $\boldsymbol{B}$ and the set of its exterior points by $\boldsymbol{T}$, with

$$\boldsymbol{I} \cup \boldsymbol{B} \cup \boldsymbol{T} = \boldsymbol{E}^3 \tag{8}$$

$$\boldsymbol{I} \cap \boldsymbol{B} = \boldsymbol{B} \cap \boldsymbol{T} = \boldsymbol{I} \cap \boldsymbol{T} = \boldsymbol{\phi} \tag{9}$$

A continuous function $f(p)$, non-negative for every p in $\boldsymbol{E}^3$, is called a defining function for a solid $\boldsymbol{S}$ if $0 < f(p) < 1$ when p belongs to $\boldsymbol{I}$, $f(p) = 1$ when p belongs to $\boldsymbol{B}$, and $f(p) > 1$ when p belongs to $\boldsymbol{T}$. For a given solid, many different defining functions can be found. For example, if $f(p)$ is a defining function for a solid $\boldsymbol{S}$, then $(f(p))^m$ (m is a positive real number) is also a defining function for $\boldsymbol{S}$.

An interesting property of a defining function is that if $f(p)$ is a defining function for a solid $\boldsymbol{S}$, then the reciprocal of the function, $(f(p))^{-1}$ is a defining function for the solid complement $\boldsymbol{S}^c$ as defined by $\boldsymbol{I}^c = \boldsymbol{T}$, $\boldsymbol{B}^c = \boldsymbol{B}$ and $\boldsymbol{T}^c = \boldsymbol{I}$ Since the defining function is always positive for all real points in $\boldsymbol{E}^3$, the complement of the function is given by $(f(p))^{-1}$. The points that have previously evaluated to less than one for $f(p)$ will thus be greater than one for $(f(p))^{-1}$, and vice versa. This means that the exterior points for $f(p)$ now become the interior points for $(f(p))^{-1}$.

The definition of a defining function indicates that for a solid $\boldsymbol{S}$ with $f(p)$ as a defining function, its surface equation is

$$f(p) = 1 \tag{10}$$

As an example, a possible defining function for a sphere with radius R and its centre at the origin of the coordinate system is

$$f(p) = (x/R)^2 + (y/R)^2 + (z/R)^2 \tag{11}$$

and $f(p) = 1$ defines the surface of the sphere.

4.2. Boolean operations (intersection and union)

One major goal of a solid modelling system is the ability to construct complex objects from simple objects. The operations required (union, intersection, and difference) are normally termed *Boolean* operations because they follow the operations defined in set theory while retaining the ability to unambiguously determine whether a point or region is "inside" or "outside" the objects.

Given n defining functions $f_1(p), f_2(p), \ldots$, and $f_n(p)$ for n objects, respectively, the defining function for the intersection of the n objects is given by

$$f^1(p) = max(f_1(p),\ f_2(p),\ \ldots,\ f_n(p)) \tag{12}$$

and the surface equation of the intersection of the n objects is given by

$$max(f_1(p),\ f_2(p),\ \ldots,\ f_n(p)) = 1 \tag{13}$$

As an example, the intersection of the three infinite slabs with defining functions

$$f_1(p) = (x/r)^2,\ f_2(p) = (y/r)^2,\ f_3(p) = (z/r)^2 \tag{14}$$

has the following surface equation

$$max((x/r)^2,\ (y/r)^2,\ (z/r)^2) = 1 \tag{15}$$

which represents the surface of a cube centred at the origin of the reference system.

Similarly, given n defining functions $f_1(p), f_2(p), \ldots$, and $f_n(p)$ for n objects, the defining function for the union of the n objects is given by

$$f^U(p) = min(f_1(p),\ f_2(p),\ \ldots,\ f_n(p)) \tag{16}$$

and the surface equation of the union of the n objects is given by

$$min(f_1(p),\ f_2(p),\ \ldots,\ f_n(p)) = 1 \tag{17}$$

4.3. Smooth approximation of intersection and union operations

Although Eqs. (13) and (17) represent the exact surfaces of the intersection and union of the n objects, they are not

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

readily manipulated and computed. To realise a smooth blending of the *n* objects into a final one, Eqs. (13) and (17) must be approximated by means of suitable functions. A certain degree of smoothing has been obtained in a particular technique for the detection of intersections of 3D objects,[29] but this method does not apply to non-convex objects. A currently widely-used method is the one reported in Ricci.[31] We use here that method. The intersection and union can be smoothly approximated as:

$$f^I(p) = (f_1^m + f_2^m + \ldots + f_n^m)^{1/m} \tag{18}$$

$$f^U(p) = (f_1^{-m} + f_2^{-m} + \ \ldots \ + f_n^{-m})^{-1/m} \tag{19}$$

where *m* is a positive real number. *m* is used to control the accuracy of the smoothing approximation and thus is called the *control parameter*. A larger *m* produces blending surfaces that cling more closely to the primitive objects. Ricci[31] proved that when $m \rightarrow \infty$, the approximations (18) and (19) give the exact description of the intersection and union respectively. These approximations have the following advantages:

1. Blending effects are primarily noticeable near surface intersections.
2. $f^I(p)$ and $f^U(p)$ are differentiable which may avoid the possible difficulties in computation due to the differentiability of the *max* and *min* functions.

The resulting approximations of the surfaces for the intersection and union of the n objects are respectively represented as:

$$(f_1^m + f_2^m + \ldots + f_n^m)^{1/m} = 1 \tag{20}$$

$$(f_1^{-m} + f_2^{-m} + \ldots + f_n^{-m})^{(-1/m)} = 1 \tag{21}$$

4.4. Useful primitive solids and the choice of m

There are many useful defining functions. The most useful ones include:

Sphere: $(x/R)^2 + (y/R)^2 + (z/R)^2 = 1$ (22)

Ellipsoid: $(x/a)^2 + (y/b)^2 + (z/c)^2 = 1$ (23)

Slabs, i.e. region bounded by two parallel planes:

$$(x/a)^2 = 1, \ (y/b)^2 = 1, \ (z/c)^2 = 1 \tag{24}$$

Circular or elliptical cylinder:

$$(x/a)^2 + (y/b)^2 = 1 \tag{25}$$

where *a*, *b*, *c*, and *R* are positive real number.

Another function which describes a broad family of easily defined primitives is the super-ellipsoid proposed in references.[26,30] The super-ellipsoid is defined by the function

$$[(x/a)^{2/e1} + (y/b)^{2/e2}]^{e2/e1} + (z/c)^{2/e2} = 1 \tag{26}$$

where *a*, *b*, and *c* define the geometric extent while e_1 and e_2 specify the shape properties. e_1 is the squareness parameter in the north-south direction; e_2 is the squareness parameter in the east-west direction. It can be seen that the super-ellipsoid can be constructed from the basic slabs given in Eq. (24).

One problem that has not been sufficiently investigated in the previous literature is the choice of the control parameter *m* in equations (20) and (21). Although Ricci[31] and Barr[26] suggested that any positive real number may be chosen as the candidate, our experience has shown that when using slabs as the basic primitives, some care must be taken. In this case, *m* must be an integer, which leads to *2m* as an even number. Some examples are given below.

Figures 9(a) and (b) show two examples, using two slabs $f_1 = x^2$, $f_2 = y^2$ as basic primitives to construct a rounded square with different orders (control parameters). In Figure 9(a), *m* has been chosen to be an integer. When using $m = 1$ to approximate the intersection of f_1 and f_2, the result is a circle. As *m* increases, the approximation to the intersection f_1 and f_2 a square with length and width being 1, gets better. It can be seen that when $m = 8$ or *16*, the approximation is very close to the square. In Figure 9(b), the values of *m* are fractions rather than integers so that *2m* is an odd number. The resulting approximate implicit function is not, as could be expected, a closed curve. Closed curve here means that the number of real circuits is limited to one and that the circuit does not extend to infinity.[27] Only in the first quadrant is it a good approximation of the intersection of f_1 and f_2. Note that in Barr.[26] Franklin and Barr,[30] the superellipsoids with different *m* are plotted out using symmetry rather than directly from the implicit function (26). This is suitable for computer graphics, but not for robot path planning. Furthermore, if *m* is chosen as a decimal so that *2m* is not an integer, then the resulting approximate implicit function is of real value only in the first quadrant:- see the figures given in Franklin and Barr.[30] The above discussion also applies to the union operation.

If, on the other hand, a circle or an ellipse is used as the primitive to construct a new object, any positive number *m* will keep the closeness of the intersection and union operation. Figures 10 and 11 give two examples. Similarly as *m* increases, the approximation gets better.

4.5. Translation and rotation

The defining functions (22) to (26) describe the solids in standard positions and orientations. It is usually necessary to translate and rotate the objects to the desired configurations. The rigid body transformations are invertible. Thus, the original inside-outside function can be used after a function inversion. For example, substituting the translation $x = (\underline{x} - a)$ into the defining function $(x/a)^2 = 1$ leads to a new defining function $[(\underline{x} - a)/a]^2 = 1$, which describes the surface of an infinite slab centred at $\underline{x} = a$ and with the same thickness of *2a*. More generally, let $\boldsymbol{M} \in \boldsymbol{R}^{3\times 3}$ denote the desired rotation matrix and $\boldsymbol{B} = [b_1, \ b_2, \ b_3]$ denote the translation vector. Then the translated and rotated solid $\underline{\boldsymbol{S}}$ is given by:

$$\underline{\boldsymbol{x}} = \boldsymbol{M}\boldsymbol{x} + \boldsymbol{B} \tag{27}$$

and the new inside-outside defining function is calculated by inverting the transformation and substituting into the old inside-outside function; i.e,

$$\underline{f}(\underline{x}, \underline{y}, \underline{z}) = f(x, y, z) \tag{28}$$

where

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

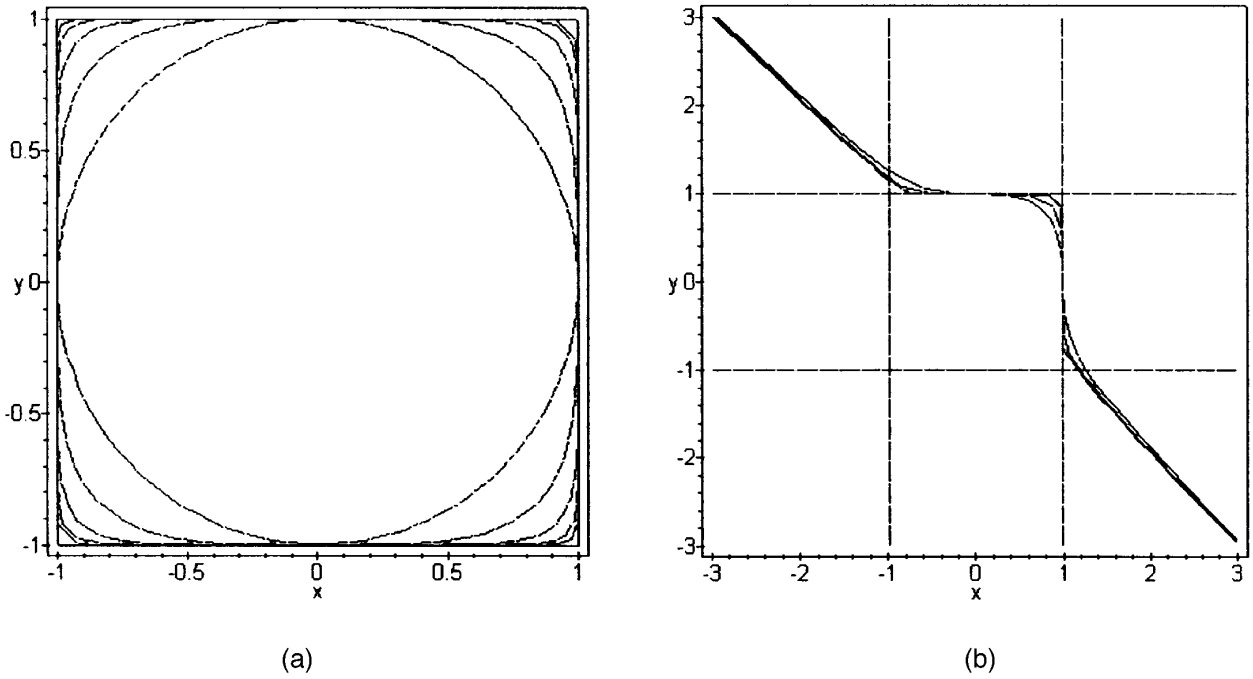


Fig. 9. Illustration of Boolean operation: Intersection $f_1 \cap f_2$. $f_1 = x^2$, $f_2 = y^2$, $((x^2)^m + (y^2)^m)^{1/m} = 1$. (a) m = 1, 2, 4, 8, and 16 from inside to outside, (b) m = 3/2, 9/2, and 33/2 from inside to outside.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \boldsymbol{M}^{-1} \begin{bmatrix} \underline{x} - b_1 \\ \underline{y} - b_2 \\ \underline{z} - b_3 \end{bmatrix} \tag{29}$$

$\boldsymbol{M}^{-1}$ is the inverse of the rotation matrix. Because the rotation matrix is always orthogonal, its inverse is the same as its transpose, i.e. $\boldsymbol{M}^{-1} = \boldsymbol{M}^T$.

5. CONVERTING THE ROBOT MOTION PLANNING PROBLEM INTO THE CONSTRAINED OPTIMISATION PROBLEM

For simplicity, and for the initial demonstration of the methods, we will consider only the motion of the UUV, limited to two dimensions. The results are readily extended to higher degrees of freedom. The vehicle is considered as a point moving in a plane, where the configuration variables are $\boldsymbol{q} = (x, y)$.

5.1. Representing the free workspace as inequality constraints on the configuration variables

The position of a point moving on a plane can be described by two configuration variables: (x, y). An obstacle in the plane can be described as

$$F_1(x, y) = 1 \tag{30}$$

Then the free space determined by this obstacle can be described by the following inequality constraint:

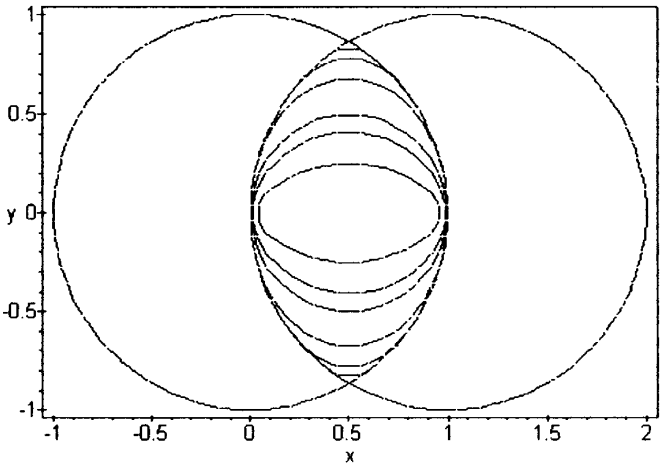


Fig. 10. Illustration of Boolean operation: Intersection $f_1 \cap f_2$. $f_1 = x^2 + y^2$, $f_2 = (x-1)^2 + y^2$. m = 0.6, 0.8, 1, 2, 5, and 25 from inside to outside.

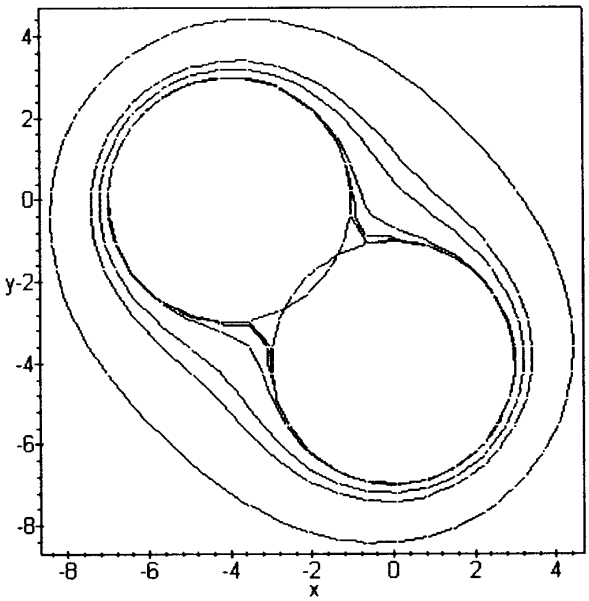


Fig. 11. Illustration of Boolean operation: *Union* $f_1 \cup f_2$. $f_1 = [(x+4)^2 + y^2]/3^2$, $f_2 = [x^2 + (y+4)^2]/3^2$. m = 8, 4, 2, 1, 0.8, and 0.5 from inside to outside.

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

$$1 - F_1(x, y) \leq 0 \tag{31}$$

Expressing the free space as the form of inequality constraint (31) is necessary to make it consistent with the requirements of MATLAB's optimisation toolbox.[37] The necessary and sufficient condition for a point which is collision-free with this obstacle is that it falls into the free space and thus that inequality (31) is satisfied.

If there are n obstacles in the workspace and the surface of the j^{th} obstacle can be described as

$$F_j(x, y) = 1, \quad j = 1, 2, \ldots, n \tag{32}$$

then the whole free space is determined by the following set of inequalities

$$1 - F_j(x, y) \leq 0, \quad j = 1, 2, \ldots, n \tag{33}$$

In addition to the obstacles, path planning should also consider the limits of the workspace. These can be described as follows

$$x_l \leq x \leq x_u, \quad y_l \leq y \leq y_u \tag{34}$$

where $x_l(y_l)$ *and* $x_u(y_u)$ are the lower and upper bounds respectively. Equations (33) and (34) are the inequality constraints required in the formulation of the non-linear programming problem for path planning.

5.2. *Design of objective function*

There are many ways to design the objective function. In the nonlinear programming problem, this function must represent some meaning of the practical problem, for example, minimum time, minimum distance, minimum energy, or minimum cost. From a mathematical viewpoint, this function must have a minimum lower bound. For the path planning problem, the goal configuration must be designed as the unique global minimum of the configuration variables. We used a quadratic function of form (35) as the objective function, with the goal configuration point (x_g, y_g) being its unique global minimum point and $min\ f(x, y) = f(x_g, y_g) = 0$.

$$f(x, y) = (x - x_g)^2 + (y - y_g)^2 \tag{35}$$

where (x_g, y_g) is the goal point of the robot. (33), (34) and (35) together form a constrained optimisation problem. If we use the initial position (x_s, y_s) as the initial estimate of the configuration variables, the optimum search for (x^*, y^*) is equivalent to searching the goal point. If the algorithm is convergent and the problem has a solution, then we will find that $x^* = x_g$, $y^* = y_g$. The main reason for choosing Eq. (35) as the objective function is that it represents the shortest distance from the start point to the goal point. Simulation results will be shown later.

In summary, the central idea for this approach is to represent the free space of the robot workspace as inequality constraints in a constrained optimisation problem using the configuration variables. The goal configuration is designed as the unique global minimum point of the objective function. The initial configuration is treated as the start point for the optimisation search. Then the numerical algorithm developed for solving the constrained optimisation problem can be applied to solve the robot motion planning problem. Every point generated using the nonlinear optimisation search method is guaranteed to be in free space and therefore is collision free.

5.3. *Algorithm implementation considerations*

In the nonlinear programming problem, early methods were based on penalty functions or barrier functions. Unfortunately penalty and barrier methods suffer from severe numerical difficulties due to possible ill-conditioned Hessian matrices in addition to the inefficiency of the sequential minimisation. Methods based on penalty and barrier functions have attracted little interest for highly nonlinear constrained problems therefore. A more direct and efficient approach, known as the sequential quadratic method (SQP) is to iterate on the basis of certain approximations to the objective and constraint functions.[34,37] The SQP method represents the state of the art in nonlinear programming and is used here to solve the robot path planning problem.

The implementation of the nonlinear optimisation presented herein is based on the constrained optimisation toolbox,[37] but some modifications were necessary. The first was the control of the output. In the reference,[37] only the final result of the variable $\boldsymbol{x}^*$ is provided. However, the important criteria for the robot path planning problem is to generate a smooth path. Therefore a new function which outputs $\boldsymbol{x}$ at every iteration has been added. The second was the change of the step length in the line search algorithm for every iteration. Recall that the principle of developing an algorithm in nonlinear programming is to minimise the number of function evaluations which represents the most efficient way for finding the optimum $\boldsymbol{x}^*$. Thus, the step length is automatically chosen as long as possible to minimise the objective function in the line search direction. The disadvantage of this strategy when applied to path planning is that it sometimes leads to a non-smooth path. We will show this point later.

5.4. *Simulation results with a quadratic function*

Example 1: A circle with center located at the point (x_c, y_c) can be described as

$$(x - x_c)^2 + (y - y_c)^2 = R^2 \tag{36}$$

where R is its radius. If any point satisfies the inequality

$$R^2 - (x - x_c)^2 - (y - y_c)^2 \leq 0 \tag{37}$$

then it must be collision-free with this circle.

The workspace considered in example 1 is surrounded by a rectangle with length 16 (− 8, 8) and width 12 (− 6, 6). It contains five obstacles distributed as in Figures 12 and 13. The obstacles are represented as circles. The boundary expressions for them are represented as follows

$$\begin{aligned} &x^2 + y^2 = 1, \quad (x - 4)^2 = 1, \quad x^2 + (y - 4)^2 = 1 \\ &(x + 4)^2 + y^2 = 1, \quad x^2 + (y + 4)^2 = 1 \end{aligned} \tag{38}$$

Then the free space can be represented as a set of the following inequality constraints

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

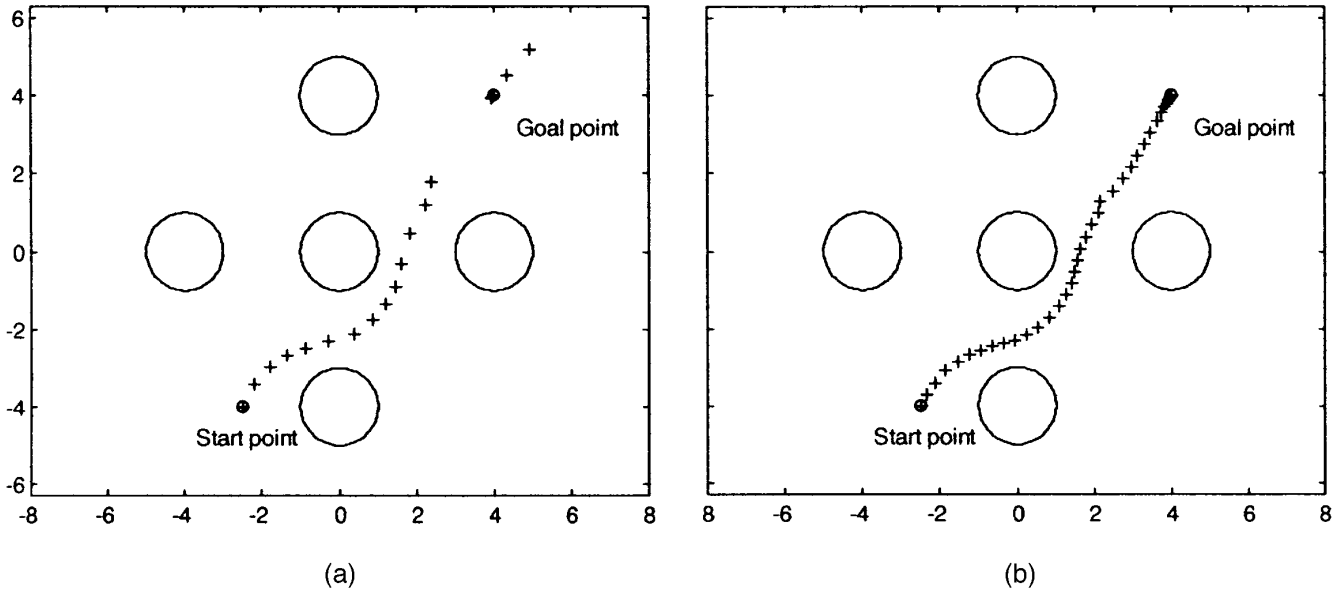


Fig. 12. (a) Illustration of the overshooting occurred due to uncontrolled step length. S = (− 2.5, − 4) and G = (4, 4), (b) Illustration of the smooth path produced using the modified algorithm. S = (− 2.5, − 4) and G = (4, 4). S = Start point, G = Goal point.

$$
\begin{aligned}
&1-(x^2+y^2)\le 0,\ 1-((x+4)^2+y^2)\le 0,\\
&1-((x-4)^2+y^2)\le 0,\quad 1-(x^2+(y-4)^2)\le 0\\
&1-(x^2+(y+4)^2)\le 0\\
&-8\le x\le 8\ \&\ -6\le y\le 6
\end{aligned}
\tag{39}
$$

The goal position is set at $\boldsymbol{q}_g = [4, 4]$. The objective function is

$$f = (x-4)^2 + (y-4)^2 \tag{40}$$

and the goal point is the only global minimum point.

Figure 12(a) shows the simulation results using the original algorithm provided in Matlab. As mentioned above, the whole path is not very smooth and overshooting occurs at the transition from point (2.399, 1.7868) to point (4.946, 5.202). So in this implementation, we set the step length to be a more appropriate size. Figure 12(b) shows the simulation result using the modified algorithm, with the same start and goal positions as those used in Figure 12(a). It is clearly shown that the path smoothness is greatly improved. For more details on the implementation of the optimisation algorithm, see reference 37.

In order to demonstrate the convergence of the problem formulation to the goal point, we have shown 9 different start points in different quadrants:- see Figures 13(a) to (i). These points represent the most difficult situations. In each case, a smooth path from the start point to the goal point has been generated. We have tested many other start points, and the results are all quite satisfactory. If the start or the goal points are located within the obstacles, the algorithm terminates, indicating which inequality was not satisfied; otherwise a smooth path can always be found.

In order to test the ability of this approach to negotiate very narrow channels among obstacles (this is hard for the potential field approach), the radius of obstacle 1 given in Eq. (38) is modified as 2.998. In this case, the narrowest gap among the obstacles is 0.002. Figure 14 illustrates the successful simulation result for the start point (− 4, − 4). Similar tests have been carried out for other start points, and the algorithm is always convergent to the goal point.

Example 2: In example 1, the distribution of the obstacles is separated in the workspace. This may represent most of the cases encountered in practice. In order to show the potential of the CSG approximation approach discussed in Section 4 for avoiding local minima and producing a smooth path maintaining some safe distance from the obstacles, we have artificially designed a critical obstacle distribution in example 2. The radii of the obstacles 4 and 5 in example 1 are expanded to 3 in example 2, while the shapes of the other obstacles are not changed. Thus, a local minima for the objective function is created in the intersection (− 2.707, − 2.707) of obstacles 4 and 5. In this way, the free space is expressed as

$$
\begin{aligned}
&1-(x^2+y^2)\le 0, \qquad 1-((x-4)^2+y^2)\le 0\\
&1-(x^2+(y-4)^2)\le 0,\ 9-((x+4)^2+y^2)\le 0\\
&9-(x^2+(y+4)^2)\le 0
\end{aligned}
\tag{41}
$$

Figures 15(a), (b) and (c) show the simulation results for different start points. In Figure 15(a), the path planning algorithm converges to the local minimum point (− 2.707, − 2.707) because this point is in the path from the start point to the goal point. However, in Figures 15(b) and (c), the algorithm successfully finds the smooth path to the goal point. Our experience indicates that the shaded region in Figure 15(d) is the forbidden region for this arrangement of objects. If the start point falls within this region, the path planning algorithm will erroneously converge to the local minimum. This experiment also indicates that if there is a local minimum point between the start and the goal points, this point is normally at an intersection between obstacles. In the nonlinear programming problem, a way to escape the local minimum is to choose another initial start point, different from the original guess. For UUV path planning, this can be applied by driving the vehicle to another location, heuristically chosen, before re-starting the motion

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

planning algorithms.

Although the strategy of choosing an immediate point away from the local minimum to escape it may be adopted, we have used another approach – the approximate Boolean operation *Union* – to deal with the local minimum. Recall that blending effects are primarily noticeable near the surface intersections. This is a very useful feature for eliminating unnecessary cusps, as shown in Figure 11. If we choose m with different values, the cusp at the intersection $(-2.707, -2.707)$ will be eliminated. As m decreases, the forbidden regions for the start point in Figure 15(d) will decrease. When $m = 0.5$, even for the most difficult start point which is in the line of the goal point and the intersection, say $(-8.-8)$, the algorithm is able to converge and produce a smooth path, as shown in Figure 16. The advantages of the Boolean operations are therefore twofold. First, they can reduce the number of the constraints, which leads to higher efficiency. Second, they can be used to overcome the local minima problem. Note that their only disadvantage is that both the start and goal points must be outside the approximate Boolean function.

5.5. Adaptive objective function for eliminating local minima

The simulation results given in Section 5.4 indicate that, even if the objective function has a unique global minimum at the goal point, for different obstacles and start point distributions the search may converge to a local minimum of

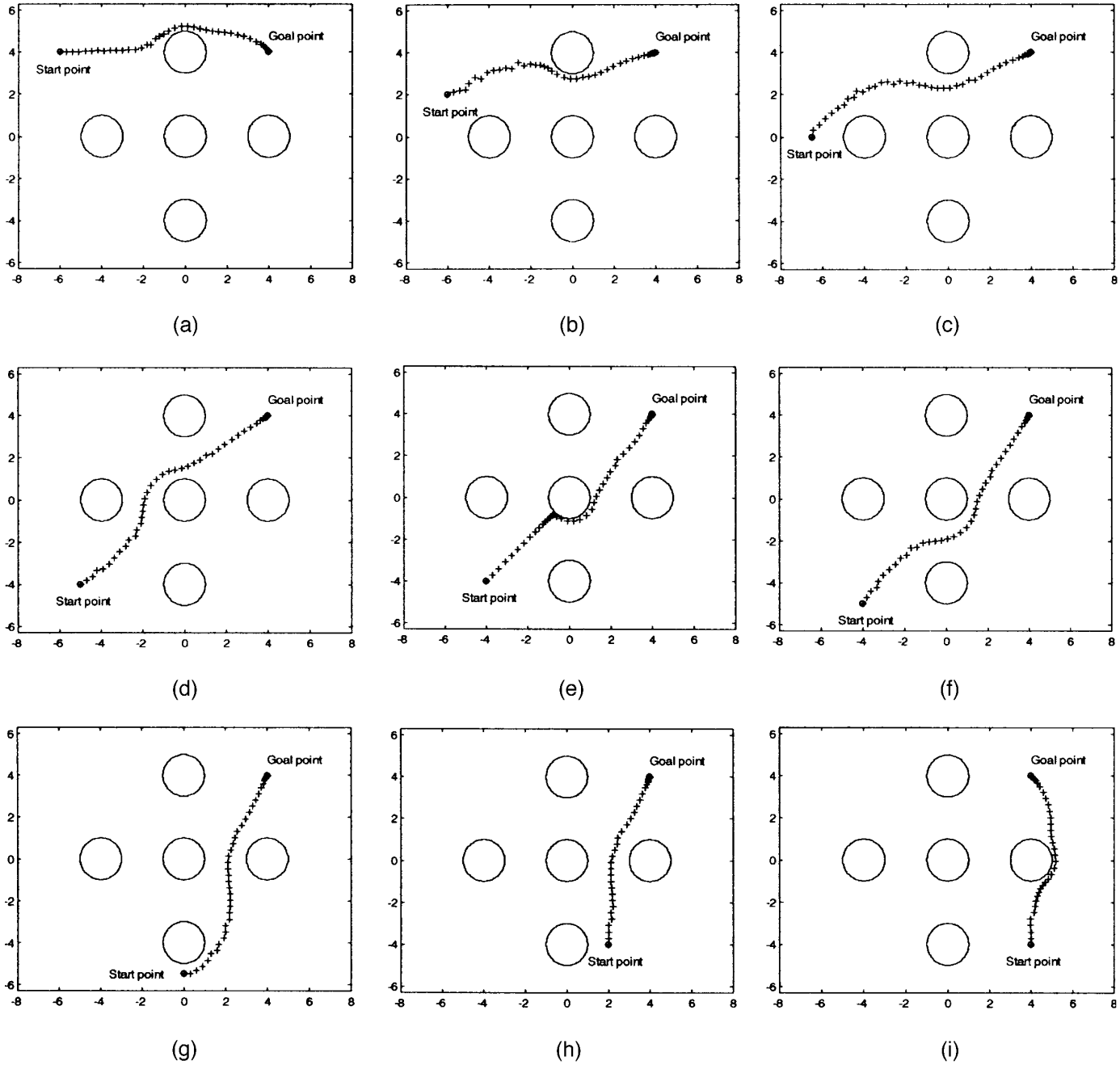


Fig. 13. Simulation results with the same goal point G = (4, 4), but different start points (a) S = (− 6, 4), (b) S = (− 6, 2), (c) S = (− 6.5, 0), (d) S = (− 5, − 4), (e) S = (− 4, − 4), (f) S = (− 5, − 4), (g) S = (0, − 5.5), (h) S = (2, − 4), and (i) S = (4, − 4). S = Start point, G = Goal point.

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

the objective function rather than the goal point. This is due to the fact that all the directions which decrease the potential function are blocked by obstacles. Using Boolean operations is one strategy for escaping from the local minima. In the following, we will develop an alternative strategy which is capable of adaptively changing the shape of the objective function.

Local minima of the objective function due to the presence of the obstacles (inequalities) are a feature widely recognised in nonlinear programming. They are different from the minimum point of the objective function without obstacles (although the global minimum of the objective function is definitely a local minimum). Local minima are produced due to the following three factors: the position of the goal configuration, the location and shape of the obstacles, and the definition of the objective function. For a given value of the objective function $f(x, y)$ in (35), Eq. (35) represents a circle in 2D space, with its centre located at the goal point. Figure 17(a) shows a contour plot with various $f(x, y)$ in 2D space. If there are obstacles in the work space like those shown in Figure 17(b), then the upper intersection point of the two circles is a local minimum point relative to the objective function (35). This indicates an important feature: local minima depend on the definition of the objective function. That is, a local minimum relative to one definition of the objective function may become a normal point (non-local minimum) relative to another. Since the position and shape of the obstacles are fixed (in the simplest case), changing the definition of objective function may lead to eliminating the local minima. As mentioned previously, designing the goal point as the unique global minimum is a necessary requirement. So a positive weighted factor w_1 is introduced to the objective function (35), resulting in the following:

$$f(x, y, w_1) = w_1(x - x_g)^2 + (1 - w_1)(y - y_g)^2 \qquad (42)$$

where w_1 is a non-negative weighted factor which satisfies

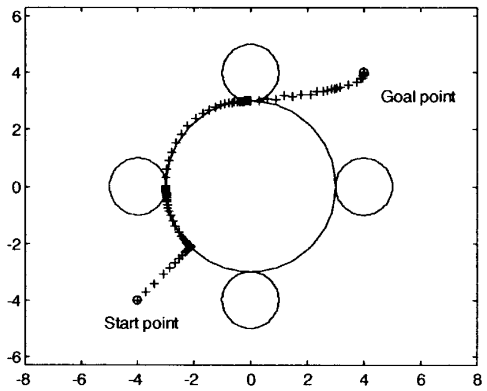


Fig. 14. Start point $(-4, -4)$ and goal point (4, 4)

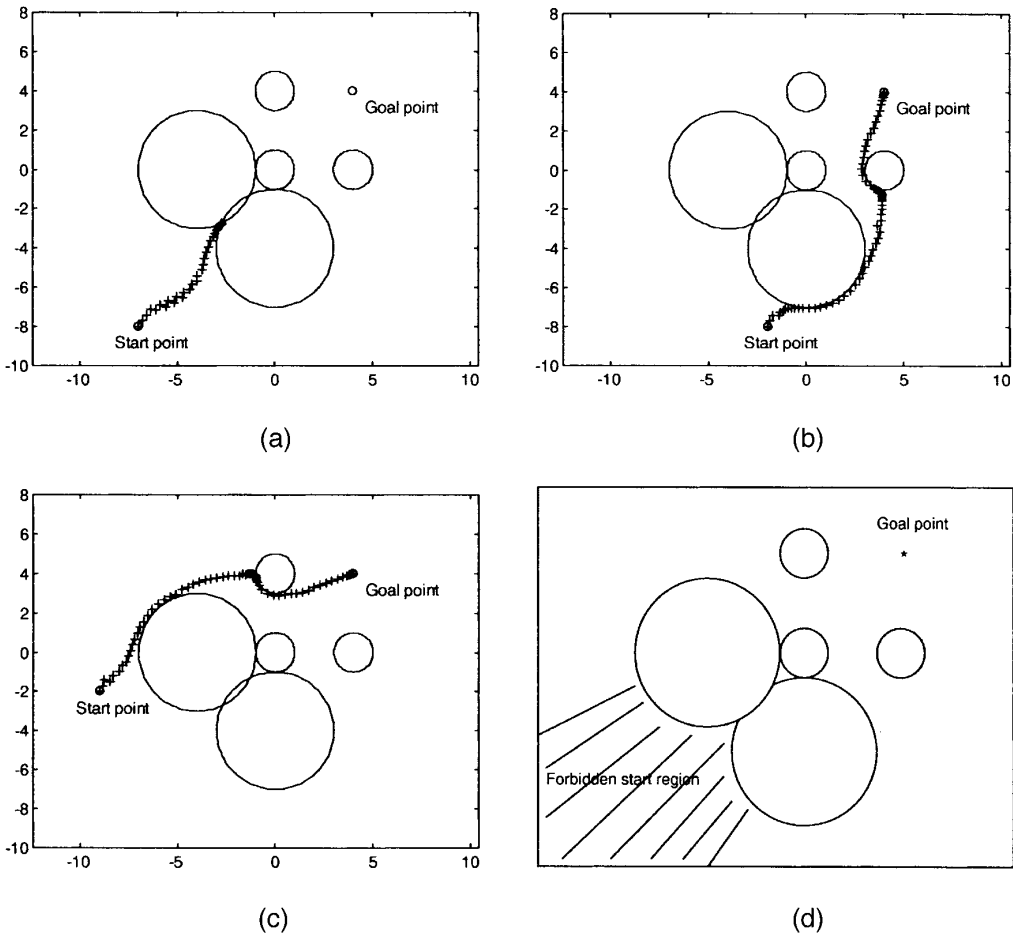


Fig. 15. (a) Start point $(-7, -8)$ and goal point (4, 4), (b) Start point $(-2, -8)$ and goal point (4, 4), (c) Start point $(-9, -2)$ and goal point (4, 4), (d) Illustration of the forbidden region for a start point, causing the optimisation search to terminate at the local minimum point, as shown in Fig. 15(a).

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

$$0 < w_1 < 1 \tag{43}$$

(35) is a special case of (42) when $w_1 = 0.5$. It is obvious that the goal $(x_g\ y_g)$ is still the unique global minimum point. If objective function (35) is viewed as a series of circles with different radii, the effect of the weighted factor w_1 is equivalent to scaling these circles simultaneously in both the x and y directions. The resulting contour plot of the objective function (42) represents a series of ellipses with their major and minor axes always being in the x and y direction. For different values of w_1 and a fixed value of the objective function $f(x, y, w_1)$ in (42), the contour plot is given in Figure 17(c). Figure 17(d) shows the contour plot when w_1 is fixed, but $f(x, y, w_1)$ varies. Although the upper intersection point of the two circles is still a local minimum in Figure 17(d), the forbidden region for the start point has decreased.

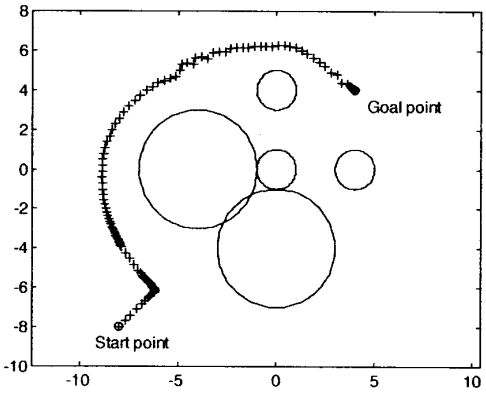


Fig. 16. Escape of the local minimum after CSG union operation is added ($m = 0.5$). Start point $(-8, 8)$ and goal point $(4, 4)$.

A more general expression of the objective function which adds a rotation degree θ into (42) has the following form

$$f(x, y, w_1, \theta) = w_1[(x - x_g)\cos\theta + (y - y_g)\sin\theta]^2 + (1 - w_1)[-(x - x_g)\sin\theta + (y - y_g)\cos\theta]^2 \tag{44}$$

For a given value of $f(x, y, w_1, \theta)$, the objective function (44) represents a general ellipse. (x_g, y_g) are the coordinates of the centre of the ellipse and are still the unique minimum point of the objective function (44), θ is the rotation angle of the major axis of the ellipse from the x-axis of the world coordinate system, and w_1 and $(1 - w_1)$ represent the lengths of the major and minor axes of the ellipse, respectively. The reason for choosing (44) as the objective function is two-fold. First, whatever w_1 and θ are, the goal point is always the unique global minimum point of the objective function and, second, with the extra variables w_1 and θ, a local minimum point for certain w_1 and θ will become the normal point for other values of w_1 and θ Figure 17(e) shows a contour plot with $f(x, y, w_1, \theta)$ and w_1 fixed, but θ varying from 0° to 90°. From Figure 17(f), we can see that the local minimum point of the intersection may be eliminated by properly changing θ.

So if we treat x, y, w_1 and θ as the variables of the Constrained Optimisation problem, then a more general formulation of the robot path planing as a Constrained Optimisation problem has been developed. (44) is the

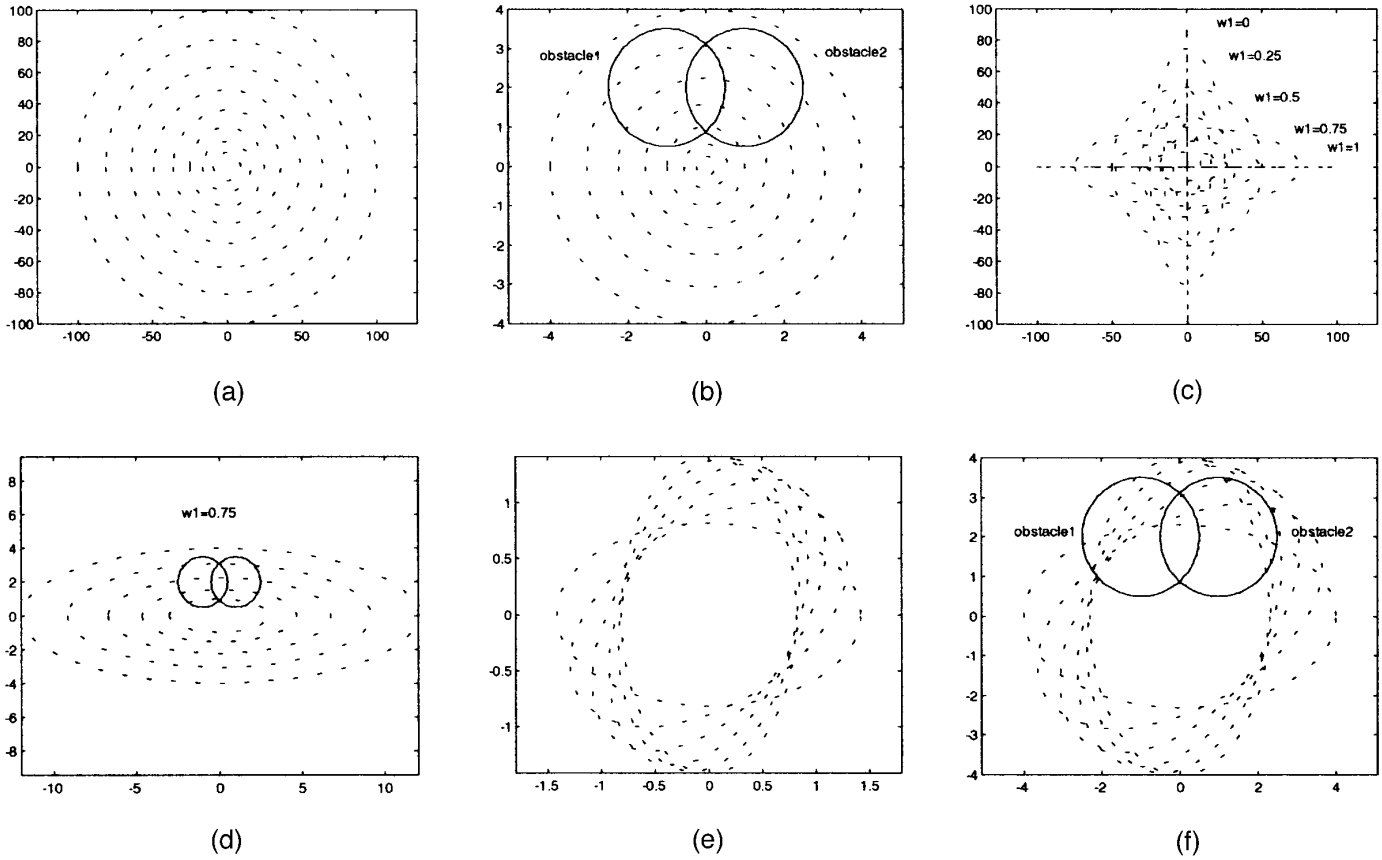


Fig. 17. Contour plots to illustrate local minima relative to different definitions of the objective function.

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

objective function, and (33), (34) and (43) are the inequality constraints. The optimisation algorithm itself decides the change of the extra variables w_1 and θ when dealing with obstacles with various locations and shapes. Therefore (44) is here referred to as the adaptive objective function. Figures 18(a) and (b) show the simulation results using the adaptive objective function, with the same obstacles as those in Figure 15.

6. CONVERTING THE ROBOT MOTION PLANNING PROBLEM INTO THE SEMI-INFINITE CONSTRAINED OPTIMISATION (SCO) PROBLEM

A path for a point mobile robot moving on a plane is a curve. There are two ways to describe a curve in a plane. The first is the implicit function which takes the form of $f(x, y) = 0$. An explicit function $y = f(x)$ is a special case of the general implicit function $f(x, y) = 0$. The second is the parametric form. The x and y are expressed as functions of an auxiliary parameter u, so that $x = x(u)$ and $y = y(u)$. In this section, we use both the implicit and the parametric forms to formulate the robot path planning as a semi-infinite constrained optimisation problem.

6.1. Undetermined coefficient approach with implicit function representation

Given the start and goal positions (x_s, y_s) and (x_g, y_g) of the robot, we use a polynomial to interpolate the values. Suppose that there exists an *nth*-order implicit polynomial which connects the start and the goal

$$y = a_0 + a_1 x + \ldots + a_n x^n = \mathbf{a} \bullet \boldsymbol{x}^T \tag{45}$$

where $\boldsymbol{a} = [a_0, a_1, \ldots, a_n]$ and $\boldsymbol{x} = [1, x, x^2, \ldots, x^n]$. It is obvious that the polynomial must satisfy the boundary conditions

$$y_s = a_0 + a_1 x_s + \ldots + a_n x_s^n \tag{46}$$

$$y_g = a_0 + a_1 x_g + \ldots + a_n x_g^n \tag{47}$$

If the obstacles are represented as a set of the inequalities like (33), then substituting (45) into (33) leads to the following inequalities

$$1 - F_j(x, a_0 + a_1 x + \ldots + a_n x^n) \le 0 \tag{48}$$

Note that inequalities (48) includes two kinds of variables, x and $\boldsymbol{a}$ where x varies over a region and $\boldsymbol{a}$ represents a point in a n-dimensional space. The robot path planning problem is to find the unknown coefficients $\boldsymbol{a}$ of the polynomial (45) which guarantee that equalities (46), (47) and inequalities (48) are simultaneously satisfied for all possible values of x. That is, if we can find the unknown coefficients $\boldsymbol{a}$ such that equalities (46), (47) and inequalities (48) are simultaneously satisfied for all possible values of x then (45) must represent a collision-free path.

There may be more than one curve which satisfies (46), (47) and (48), we choose the one which minimises the following objective function

$$f(\boldsymbol{a}) = \sum_{i=0}^{n} \boldsymbol{a}_i^2 \tag{49}$$

It is obvious that (46), (47), (48) and (49) together form a semi-infinite constrained optimisation problem of the form (2) with a_i being the unknown variables and x being the only one independent variable.

6.2. Undetermined coefficient approach with parametric function representation

Similarly, given the start and goal positions (x_s, y_s) and (x_g, y_g) of the robot, suppose that there exists an *nth*-order parametric polynomial which connects the start and the goal

$$x = b_0 + b_1 u + \ldots + b_n u^n = \boldsymbol{b} \bullet \boldsymbol{u}^T \tag{50}$$

$$y = c_0 + c_1 u + \ldots + c_n u^n = \boldsymbol{c} \bullet \boldsymbol{u}^T \tag{51}$$

where $\boldsymbol{b} = [b_0, b_1, \ldots, b_n]$, $\boldsymbol{c} = [c_0, c_1, \ldots, c_n]$, $\boldsymbol{u} = [1, u, u^2, \ldots, u^n]$, and $0 \le u \le 1$. $u = 0$ and $u = 1$ correspond to the initial and goal positions of the robot, respectively. Thus

$$b_0 = x_s, \; c_0 = y_s \tag{52}$$

$$\sum_{i=0}^{n} b_i = x_g, \quad \sum_{i=0}^{n} c_i = y_g \tag{53}$$

If the obstacles are represented as a set of inequalities like

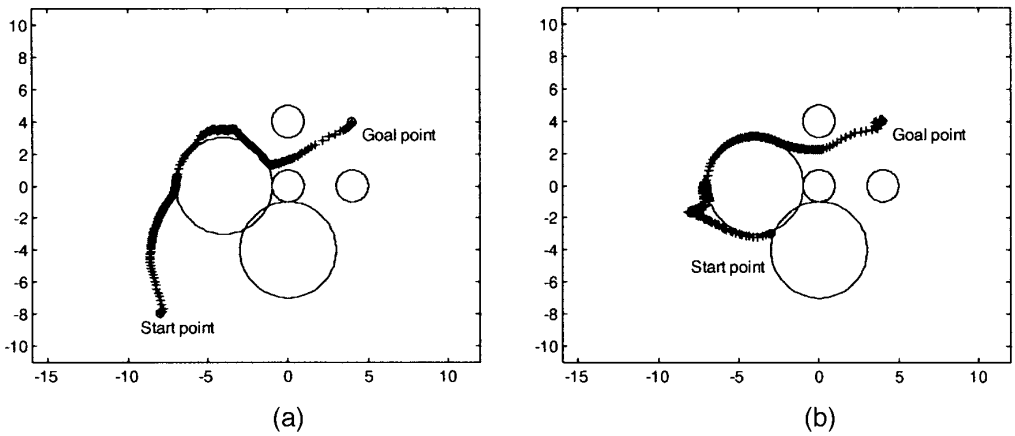


Fig. 18. Simulation results with the adaptive objective function (a) Start point = (− 8, − 8) (b) Start point = (− 3, − 3).

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

(33), then substituting (50) and (51) into (33) gives the following inequalities

$$1 - F_j(b_0 + b_1u + \ldots + b_nu^n, c_0 + c_1u + \ldots + c_nu^n) \leq 0 \quad (54)$$

(54) represents a set of inequalities with b_i and c_i being the unknown variables and the parameter u as the independent variable. If we can find the unknown coefficients $\boldsymbol{b}$ and $\boldsymbol{c}$ such that equalities (52), (53) and inequalities (54) are simultaneously satisfied for all possible values of u, then (50) and (51) must represent a collision-free path.

The objective function in this formulation is to minimise the sum of the square of the coefficients $\boldsymbol{b}$ and $\boldsymbol{c}$

$$f(\boldsymbol{b}, \boldsymbol{c}) = \sum_{i=0}^{n} (b_i^2 + c_i^2) \quad (55)$$

In this case (52), (53), (54) and (55) together form a semi-infinite constrained optimisation problem of the form (2) with b_i and c_i being the unknown variables and u being the only one independent variable.

In summary, the central idea for the undetermined coefficient approach is as follows: first the interpolation technique is used to fit the initial and final configuration requirements, second the free space of the robot workspace is represented as inequality constraints with the undetermined coefficients and one independent variable, and finally an objective function of the undetermined coefficients is designed. The above three parts form a semi-infinite constrained optimisation problem. The numerical algorithm developed for solving semi-infinite constrained optimisation problem can be applied to solve the undetermined coefficients. Once a solution is found, the interpolation function must be a collision-free path connecting the initial and the goal points.

6.3. Simulation results

6.3.1. Undetermined coefficient approach with implicit function representation. In this simulation, the obstacles are the same as those in the example 1 and therefore they are represented by inequalities (39). The start and goal positions are $(x_s, y_s) = (-4, -4)$ and $(x_g, y_g) = (4, 4)$, respectively. The objective function is the same as (49). The order of the parametric polynomial has been chosen as $n = 7$. The procedure for robot path planning using this approach is to first randomly choose a set of estimated coefficients of $\boldsymbol{a}$, denoted as $\boldsymbol{a}_{ie}$, and then run the semi-infinite constrained optimisation algorithm. The output of the algorithm is donated as $\boldsymbol{a}_{f*}$. The simulation results are shown in Figures 19(a), (b), (c), and (d) for different $\boldsymbol{a}_{ie}$. The dashed curve represents the robot path from the estimated coefficients $\boldsymbol{a}_{ie}$ using (45), and the solid curve represents the generated collision-free path from the final coefficients $\boldsymbol{a}_{f*}$ using (45). Note that it is possible that the curve resulting from the initial estimate $\boldsymbol{a}_{ie}$ collides with the obstacles, as shown in Figure 19.

In Figures 19(a), (b), and (c), although the estimated coefficients $\boldsymbol{a}_{ie}$ are different and the resulting final coefficients $\boldsymbol{a}_{f*}$ are also different, the shape of the collision-free curves is quite similar. In Figure 19(d), the dashed curve from coefficients $\boldsymbol{a}_{ie}$ are far away from the start and goal points, the generated collision free path shows a little unexpected oscillation.

6.3.2. Undetermined coefficient approach with parametric function representation. In this simulation, the obstacles are still represented by inequalities (39). The start and goal positions are $(x_s, y_s) = (-4, -4)$ and $(x_g, y_g) = (4, 4)$, respectively. The objective function is chosen as (55). The order of the parametric polynomial is chosen as $n = 7$. The procedure for robot path planning using this approach is to first randomly choose a set of estimated coefficients for $\boldsymbol{b}$ and $\boldsymbol{c}$, denoted as $\boldsymbol{b}_{ie}$ and $\boldsymbol{c}_{ie}$ and then run the semi-infinite constrained optimisation algorithm. The simulation results are shown in Figures 20(a) and (b). The dashed curve represents the robot path from the estimated coefficients $\boldsymbol{b}_{ie}$ and $\boldsymbol{c}_{ie}$ using (50) and (51), and the solid curve represents the generated collision-free path from the final coefficients $\boldsymbol{b}_f$* and c_f* using (50) and (51). In Figure 20(a) the initial estimate for the coefficients of $\boldsymbol{b}$ and $\boldsymbol{c}$ in (50) and (51) are chosen as $\boldsymbol{b}_{ie} = [-4, 2, 2, 1, 2, 1, 0, 0]$ and $\boldsymbol{c}_{ie} = [-4, 1, 1, 2, 1, 2, 1, 0]$ which satisfy the two equality constraints (52) and (53). This can be seen in Figure 20(a) from the fact that the dashed curve connects the start and the goal points. The final coefficients after running the semi-infinite constrained optimisation algorithm are $\boldsymbol{b}_{f*} = [-4, 2.1586, 1.7104, 1.3402, 1.0344, 0.7819, 0.5734, 0.4012]$ and $\boldsymbol{c}_{f*} = [-4, 0.3090, 0.6770, 0.9809, 1.2319, 1.4392, 1.6104, 1.7518]$.

In Figure 20(b) the initial estimate for the coefficients of $\boldsymbol{b}$ and $\boldsymbol{c}$ in (50) and (51) are chosen as $\boldsymbol{b}_{ie} = [-4, 3, 2, 1, 1, 0, 1, -1]$ $\boldsymbol{c}_{fe} = [-4, 1, 4, 2, 1, -1, 1, 1]$. In this case only one equality constraint of (52) is satisfied. The final coefficients are $\boldsymbol{b}_{f*} = [-4, 0.3090, 0.6770, 0.9809, 1.2319, 1.4392, 1.6104, 1.7518]$, $\boldsymbol{c}_{f*} = [-4, 2.1586, 1.7104, 1.3402, 1.0344, 0.7819, 0.5734, 0.4012]$. Both of them have given satisfactory results, as shown in Figures 20(a) and (b).

7. CONCLUSIONS

In this paper, two systematic approaches to motion planning for subsea vehicle using the optimisation techniques have been presented. We have shown the principle of converting the path planning problem into the standard Constrained Optimisation and Semi-infinite Constrained Optimisation problems. They are different from the traditional approaches both in the representation of the objects and in the design of the search algorithm. In traditional approaches, polygons are normally used for object representation and the octree search for finding the collision-free path. Although it is possible to represent virtually any surface with sufficient numbers of polygons, it is sometimes more convenient to use the basic defining functions which lead to a more compact representation and easier manipulation than polygons. CSG and the Boolean operations have made it possible to represent a wide range of objects such as cylinders, spheres and even polyhedra (polyhedra may be represented by the slabs and the transformation technique) as a single algebraic function using the basic defining functions. The problem of slow search speed with large numbers of constraints may be overcome by the approx-

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

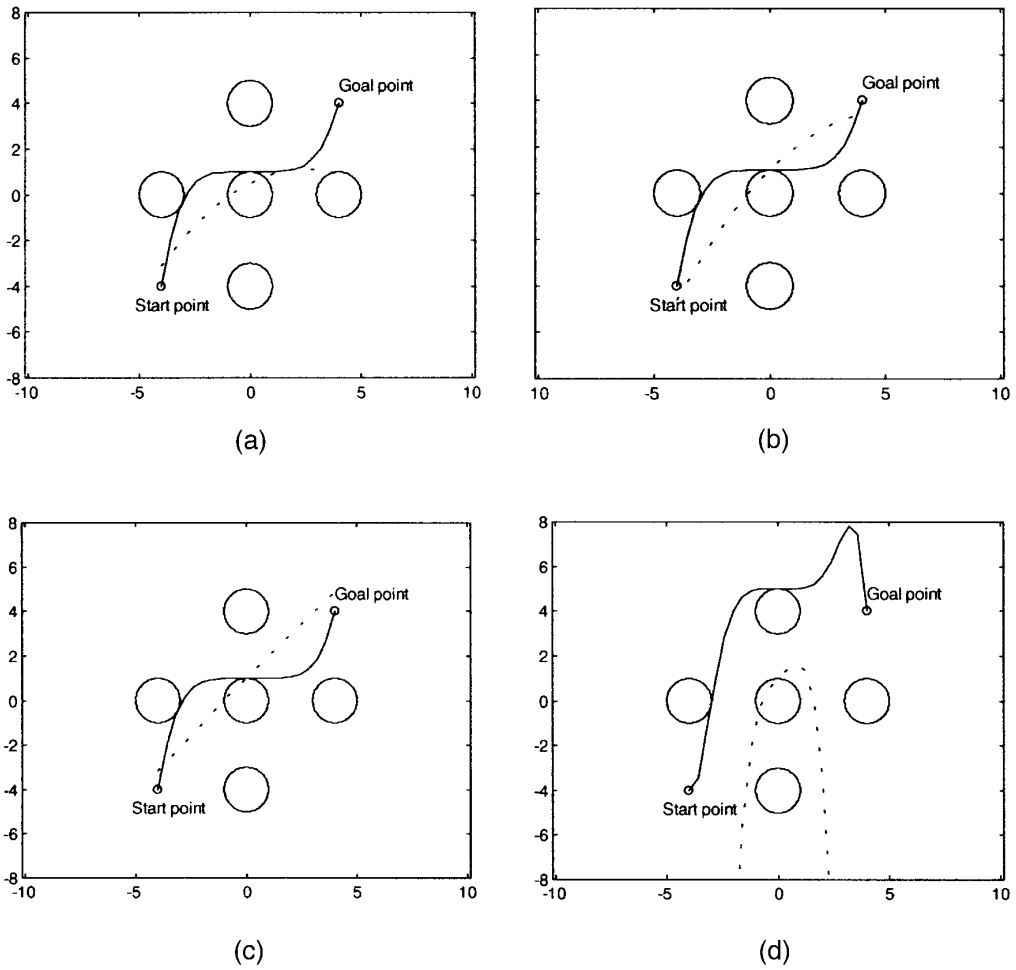


Fig. 19. Simulation results using Semi-infinite Constrained Optimisation (SCO) approach with implicit function. Estimated robot path (dashed curve) and generated collision-free path (solid). (a) $\boldsymbol{a}_{ie}$ = [0.50, 0.50, −0.10, 0, 0, 0, 0, 0], $\boldsymbol{a}_{f^*}$ = [1.0000, 0.0000, −0.0002, 0.0007, −0.0014, 0.0042, −0.0002, 0.0000], (b) $\boldsymbol{a}_{ie}$ = [1.00, 1.00, −0.10, 0, 0, 0, 0, 0], $\boldsymbol{a}_{f^*}$ = [1.0000, 0.0000, −0.0002, 0.0007, −0.0015 0.0042, −0.0002, 0.0000], (c) $\boldsymbol{a}_{ie}$ = [1.00, 1.00, −0.01, 0, 0, 0, 0, 0], $\boldsymbol{a}_{f^*}$ = [1.0000,0.0001, −0.0002, −0.0001, −0.0042, 0.0035, 0.0000, 0.0000], (*d*) $\boldsymbol{a}_{ie}$ = [1, 1, −0.2, 0.3, −0.5, 0, 0, 0], $\boldsymbol{a}_{f^*}$ = [2.1586, 1.7104, 1.3402, 1.0344, 0.7819, 0.5734 0.4012, 0.3090].

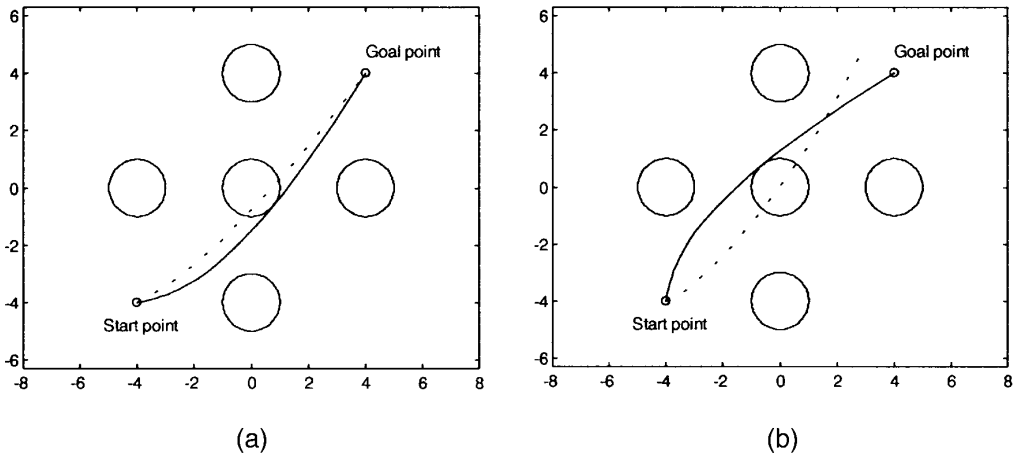


Fig. 20. Simulation results using Semi-infinite Constrained Optimisation (SCO) approach with parametric function. Estimated robot path (dashed curve) and generated collision-free path (solid). (a) $\boldsymbol{b}_{ie}$ = [−4, 2, 2, 1, 2, 1, 0, 0], $\boldsymbol{c}_{ie}$ = [−4, 1, 1, 2, 1, 2, 1, 0]. $\boldsymbol{b}_{f^*}$ = [−4, 2.1586, 1.7104, 1.3402, 1.0344, 0.7819, 0.5734, 0.4012], $\boldsymbol{c}_{f^*}$ = [−4, 0.3090, 0.6770, 0.9809, 1.2319, 1.4392, 1.6104, 1.7518]. (b) $\boldsymbol{b}_{ie}$ = [−4, 3, 2, 1, 1, 0, 1, −1], $\boldsymbol{c}_{fe}$ = [−4, 1, 4, 2, 1, -1, 1, 1]. $\boldsymbol{b}_{f^*}$ = [−4, 0.3090, 0.6770, 0.9809, 1.2319, 1.4392, 1.6104, 1.7518], $\boldsymbol{c}_{f^*}$ = [−4, 2.1586, 1.7104, 1.3402, 1.0344, 0.7819, 0.5734, 0.4012].

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press

imate Boolean operations. The mature search techniques developed in nonlinear programming enable us to concentrate on the new problems encountered in its application to the path planning problem.

The advantages of the approaches are that mature techniques developed in nonlinear programming theory which guarantee convergence, efficiency and numerical robustness can be directly applied to the robot path planning problem. We have analysed the factors which may result in local minima, the main issues for the potential field approach, and discussed the difficulty in avoiding them. The Constrained Optimisation approach with an adaptive objective function has the following advantages:

(i) The goal point is guaranteed to be the only global minimum of the objective function.
(ii) Local minimum problems arising from the potential field approach can be avoided.
(iii) The standard search techniques which have been developed for more than thirty years in the nonlinear programming field can be used.
(iv) The approach is suitable for on-line task planning.

From the viewpoint of efficiency, the CO approach is much better than the SCO formulation. However, the second formulation does not suffer the local minima problem. This is its main advantage. On the other hand, an issue which deserves further investigation for the second approach is the choice of order of the polynomial which determines the degrees of freedom left after the interpolation to the initial and the goal requirements. Currently we used only one polynomial to interpolate the initial and the goal points. In order to have enough degrees of freedom after interpolation, a higher order (normally more than 4) is used. However, as has been seen, a polynomial of a higher order may cause some unexpected oscillation. A possible solution could be the use of lower order piece-wise polynomials (for instance, cubic). These are topics for future research.

The research carried out in this paper has indicated that robot path planning can be formulated as different optimisation problems. Although the fundamentals for both the nonlinear programming theory and the CSG as well as the Boolean operations have existed for many years, they have not attracted enough attention for such applications. The context presented in this paper covers a wide range of subjects such as robot kinematics, CAD, CAM, Computer Graphics and nonlinear programming theory, and a basic framework has been developed. Our treatment is consistent. The study presented in this paper has shown its great potential as an on-line motion planner.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support of ESPRIT III Basic Research Action #8972 UNION — Underwater Intelligent Operation and Navigation, 1994–1996.

References

1. M.W. Dunnigan, D.M. Lane, A.C. Clegg and I. Edwards, "Hybrid position/force control of a hydraulic underwater manipulator", *IEE Proc. Part D Control Theory and Applications* **143**, 145–151 (1996).
2. D.M. Lane, M.W. Dunnigan, A.W. Quinn and A.C. Clegg "Motion planning and contact control for a tele-assisted hydraulic underwater robot", *J. Autonomous Robots* **3**, 233–251 (1996).
3. D.M. Lane and P.J.K. Knightbridge, "Task planning and world Modelling for supervisory control of robots in unstructured environments", *Proc. IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, (1995) pp. 1880–1885.
4. Y. Petillot, I. Tena Ruiz, D.M. Lane, Y. Wang, M. Trucco, and N. Pican, "Underwater vehicle path planning using a multibeam forward looking sonar", *Proc. of IEEE/OES OCEANS 98*, Nice, France, (1998) pp. 1194–1199.
5. Yongji Wang and D.M. Lane "Subsea vehicle path planning using nonlinear programming and constructive solid geometry", *IEE Proc. Part D, Control Theory and Applications* **144**, 143–152 (1997).
6. R.A. Brooks and T. Lozano-Perez, "A subdivision algorithm in configuration space for findpath with rotation", *IEEE Trans. on Systems, Man and Cybernetics* **15**, 224–233 (1985).
7. M.W. Chen and A.M.S. Zalzala "A genetic-based approach to robot motion planning considering path safety and moving obstacles", *Proc. of the European Chinese Automation Conference*, London, (1995), pp. 143–148.
8. T.C. Hu, A.B. Kahng and G. Robins, "Optimal robust path planning in general environment", *IEEE Trans. Robotics and Automation* **9**, 755–774 (1993).
9. H.P. Huang and P.C. Lee, "A real-time algorithm for obstacle avoidance of autonomous mobile robots", *Robotica* **10**, Part 3, 217–227 (1992).
10. Y.K. Hwang and N. Ahuja, "A potential field approach to path planning", *IEEE Trans. on Robotics and Automation* **8**, 23–32 (1992).
11. O. Khatib, "Real-time obstacle avoidance for manipulator and mobile robots", *Int. J. Robotics Research* **5**, 90–98 (1986).
12. J.C. Latombe, *Robot Motion Planning* (Kluwer Academic Publishers, Boston, 1991).
13. Z.X. Li, and T.D. Bui, "Robot path planning and navigation using fluid model", *Proc. of Second Int. Conf. on Automation, Robotics and Computer Vision (ICARCV)*, Singapore, (1992) pp. RO–10.3.1.
14. T. Lozano-Perez, "Spatial planning: A configuration space approach", *IEEE Trans. on Computers* **32**, 108–120 (1983).
15. J. Lumelsky, "A comparative study on path length performance of maze-searching and robot motion planning", *IEEE Trans. on Robotics and Automation* **7**, 57–66 (1991).
16. M. Schlemmer and G. Gruebel, "Real-time collision-free trajectory optimisation of robot manipulators via semi-infinite parameter optimisation", *Int. J. Robotics Research* **17**, 1013–1021 (1998).
17. J.T. Schwartz and M. Sharir, "On the 'piano movers' problem: I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barrios", *Communications on Pure and Applied Mathematics* **XXXVI**, 375–398 (1983).
18. J.T. Schwartz and M. Sharir, "On the 'piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers", *Int. J. Robotics Research* **2**, 46–75 (1983).
19. S. Sundar and S. Shiller, "Optimal obstacle avoidance based on the HJB equation", *IEEE Trans. on Robotics and Automation* **13**, 305–310 (1997).
20. Yongji Wang, J.A. Linnett and J.W. Roberts, "Kinematics, kinematic constraints and path planning for wheeled mobile robots", *Robotica* Part 5, **12,** 391–400 (1994).
21. Yongji Wang, "Kinematics, motion analysis and path planning for four kinds of wheeled mobile robots" *Ph.D Thesis*, (Department of Mechanical Engineering, Edinburgh University, 1995).
22. Yongji Wang, "Nonholonomic motion planning: a polynomial fitting approach", *1996 IEEE Int. Conf. on Robotics and Automation*, Minnesota, USA, (1996) pp. 2956–2961.

23. Yongji Wang, "A note on solving the find-path problem by good representation of free space", *IEEE Trans. on Systems, Man and Cybernetics (Part B)* **27**, 723–725 (1997).
24. Yongji Wang and M.P. Cartmell, "Autonomous vehicle parallel parking design using function fitting approaches", *Robotica* **16**, Part 2, 159–170 (1998).
25. Y. Zhang and K.P. Valavanis, "A 3-D potential panel method for robot motion planning", *Robotica*, **15**, Part 4, 421–434 (1997).
26. H. Barr, "Superquadrics and angle-preserving transformations", *IEEE Computer Graphics and Applications* **1**, 11–23 (1981).
27. J.L. Blechschmidt and D. Nagasuru, "The use of algebraic functions as a solid modelling alternative: an investigation", *Proc. of the 16th ASME Design Automation Conf.*, Chicago, USA (1990) pp. 33–41.
28. H. Chiyokura, *Solid Modelling with Designbase* (Addison-Wesley Publishing Limited, New York, 1988).
29. P.G. Comba, "A procedure for detecting intersections of three-dimensional objects" *JACM* **15**, 354–366 (1968).
30. W.R. Franklin and A.H. Barr, "Faster calculation of superquadric shapes", *IEEE Computer Graphics and Applications* **1**, 41–57 (1981).
31. A. Ricci, "A constructive geometry for computer graphics", *The Computer Journal* **16**, 157–160 (1973).
32. R. Fletcher, *Practical Methods of Optimisation* (John Wiley & Sons., New York, 1987).
33. P.E. Gill, W. Murray and M. H. Wright, *Practical Optimisation* (Academic Press, London, 1981).
34. S.P. Han, "A globally convergent method for nonlinear programming" *J. Optimisation Theory and Applications* **22**, 297–235 (1977).
35. D.G. Luenberger, *Linear and Nonlinear Programming* (Addison-Wesley Publishing Company, 2nd ed. London, 1984).
36. M.J.D. Powell, "A fast algorithm for nonlinear constrained optimisation calculation" *Numerical analysis*, (ed., G.A. Watson) *Lecture Notes in Mathematics* (Springer Verlag, Berlin, 1978). **Vol 630**.
37. The Math Works Inc., *MATLAB, Optimisation Toolbox User's Guide,* (June, 1993).

https://doi.org/10.1017/S0263574799002015 Published online by Cambridge University Press