Robotica (2004) volume 22, pp. 173–187. © 2004 Cambridge University Press
DOI: 10.1017/S0263574703005472 Printed in the United Kingdom

A real-time path planning approach without the computation of Cspace obstacles

Yongji Wang*, Liu Han*, Mingshu Li*, Qing Wang*, Jinhui Zhou* and Matthew Cartmell†

(Received in Final Form: May 22, 2003)

SUMMARY

An important concept proposed in the early stage of robot path planning field is the shrinking of the robot to a point and meanwhile expanding of the obstacles in the workspace as a set of new obstacles. The resulting grown obstacles are called the Configuration Space (Cspace) obstacles. The find-path problem is then transformed into that of finding a collision free path for a point robot among the Cspace obstacles. However, the research experiences obtained so far have shown that the calculation of the Cspace obstacles is very hard work when the following situations occur: 1. both the robot and obstacles are not polygons and 2. the robot is allowed to rotate. This situation is even worse when the robot and obstacles are three dimensional (3D) objects with various shapes. Obviously a direct path planning approach without the calculation of the Cspace obstacles is strongly needed. This paper presents such a new real-time robot path planning approach which, to the best of our knowledge, is the first one in the robotic community. The fundamental ideas are the utilization of inequality and optimization technique. Simulation results have been presented to show its merits.

KEYWORDS: Real-time system; Robotics; Autonomous underwater vehicle; Subsea vehicle; Path planning; Obstacle avoidance; Non-linear programming; Semi-infinite constrained optimization.

1. INTRODUCTION

An autonomous robot (for example, an Unmanned Underwater Vehicle extensively used for the inspection, maintenance and repair of offshore structures in both deep and shallow water, (see Figure 1) is a typical real-time system which has strict time constraints.[1–7] Such a real-time system normally consists of the following tasks: 1. Dynamic plant identification; 2. Low level control law design; 3. Real-time sensor fusion or computer vision signal processing for environmental map building; 4: Path planning based on the results obtained in step 3; and 5: High level mission management or control.[8–12] In this paper we concentrate on the path planning task 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 efficiently 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 and collision-free path from the initial configuration to the goal configuration for the vehicle.

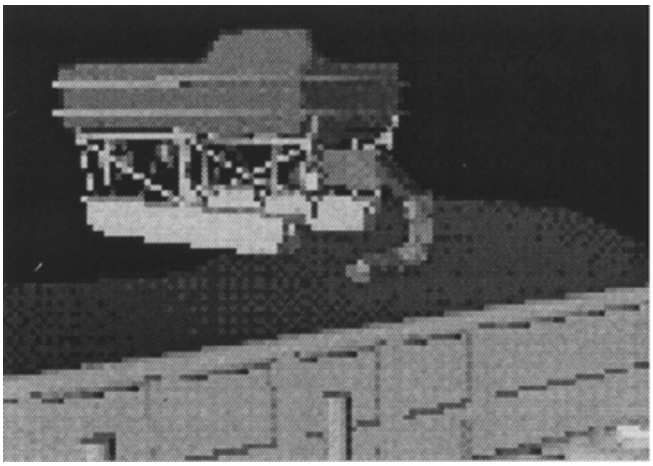
Robot path planning has been an active field during the past twenty years. A significant and varied effort has been made on this seemingly simple, but in fact very complicated find-path problem.[13–34] Path planning problem essentially deals with how to move a 3D object (robot) among another set of 3D objects (obstacles), satisfying various constraints.[22,25,28,29] Various methods for dealing with the basic find-path problem and its extensions, such as Vgraph,[15] Voronoi diagram,[13] exact cell decomposition,[14] approximate cell decomposition,[13] potential field approach,[6,17,18,34] inequality-optimization-based approach,[1,26,27] and other approaches[16,21,24] have been developed. A systematic discussion on these methods can be found in references.[8,13]

In any robot path planning method, robot and obstacle representation is the first thing to be considered. Historically the Constrained Optimization and Constructive Solid Geometry (COCSG) method was first proposed in Wang et. al.[1] and the assumption that each obstacle in the workspace can be represented by one inequality is made. The mathematical foundations for the Constructive Solid Geometry (CSG), the Boolean operations and the approximation techniques were developed to represent the free space of the robot as a set of inequalities. The fundamental ideas used include: (i) the free Cspace of the robot is represented as a set of inequality constraints using the vehicle configuration variables; (ii) the goal configuration is designed as the unique global minimum point of the objective function and the initial configuration is treated as the start point for the spatial search; (iii) the numerical algorithm developed for solving nonlinear programming problem is applied to solve the robot motion planning problem and every immediate point generated in this way guarantees that it is in the free Cspace and therefore is collision free.

An important concept proposed in the early stage of robot path planning field is the shrinking of the robot to a point

* Laboratory for Internet Technologies, Institute of Software, Chinese Academy of Sciences, P.O. Box 8718, Beijing 100080 (China).
E-mail: {ywang, liuhan, wq, jinhui}@intec.iscas.ac.cn, mingshu@admin.iscas.ac.cn
† Department of Mechanical Engineering, Centre for Systems and Control, James Watt (South) Building, University of Glasgow, Glasgow, G12 8QQ (U.K.).
E-mail: m.cartmell@mech.gla.ac.uk

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press



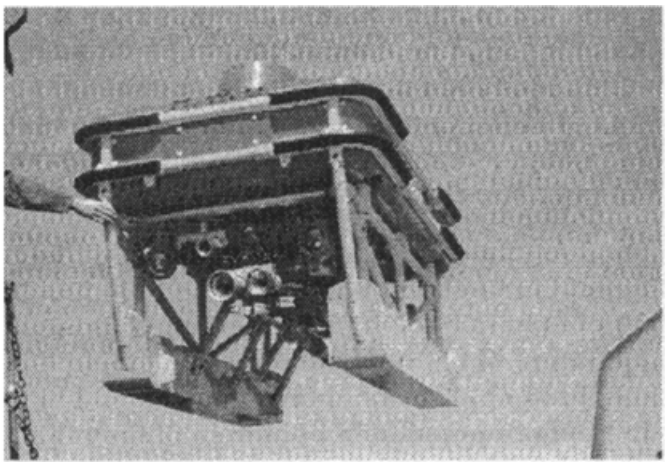


Fig. 1. ANGUS 003, an unmanned underwater vehicle.

and meanwhile the expanding of the obstacles in the workspace as a set of new obstacles. The resulting grown obstacles are called the Configuration Space (Cspace) obstacles. The find-path problem is then transformed into that of finding a collision free path for a point robot among the Cspace obstacles. This idea was first popularized by Lozano-Perez[15] in the Artificial Intelligence Lab, MIT as a basis of the spatial planning approach for the find-path and find-place problems and then extended by Latombe[13] as a basis of all motion planning approaches suitable for a point robot. However, the research experiences obtained so far have shown that the calculation of the Cspace obstacles is difficult when the following situations occur. 1. Both the robot and obstacles are not polygons and 2. The robot is allowed to rotate. This situation is becoming even worse when the robot and obstacles are 3D objects with various shapes[35–38]. It can be observed that all of the aforementioned approaches developed so far are based on the Cspce obstacles; therefore they are only suitable for the unrealistic situation. Obviously, a direct path planning approach without the calculation of the Cspace obstacles is strongly needed. This paper presents such a new real-time robot path planning approach which, to our best knowledge, is the first one in the robotic community.

Note as shown later, the ideas of inequality and optimization proposed by Wang[1,26,27] are still useful and will be used in this paper. They enable us to develop a new approach without the need to calculate the Cspace obstacles. Therefore the work reported here is, in certain way, a natural extension of the previous work.

In this paper we will present the principle of the new real-time robot path planning. The points outside the obstacles in the 3D workspace are represented as implicit inequalities and the points on the boundary of a 3D robot are expressed in the form of a parametric function. A sufficient and necessary condition for a collision free path for the robot and the obstacles is then derived in the form of a set of inequalities that lead to the use of efficient search algorithms.

This paper is organized as follows. Section 2 gives a brief description of the formulations for both the constrained optimization (CO) and semi-infinite constrained optimization (SICO). In Section 3, obstacle and robot representation is presented. The Constructive Solid Geometry technique for representing obstacles as inequalities and Euler angle conversion for the representation of a 3D robot are also given. In Section 4, we investigate how to convert the robot path planning problem into the SICO problem and a sufficient and necessary condition for a collision-free path for a 3D robot among a set of 3D obstacles is developed. Simulation results are the content of Section 5. Finally conclusions are given in Section 6.

2. THEORETICAL BACKGROUND

Optimization concerns the minimization or maximization of a function subject to different types of constraints (equality or inequality). The function is referred to as the objective function. Optimization 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 optimization 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. Optimization techniques have now become a mature subject. In the following, we give a brief introduction to optimization problems because the emphasis of this paper will be focused on how to convert the path planning problem into a standard semi-infinite constrained optimization problem, rather than on the details of the nonlinear programming theory which are easily found.[39–45]

There are mainly four different types of optimization problem: *Linear Programming*, *Unconstrained Problems*, *Constrained Problems* and *Seminfinite Constrained Problems* see Table I. The last three parts together comprise the subject of *Nonlinear Programming*. When applying optimization 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.

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

Table I. Nonlinear minimization problems*.

Type	Notation
Unconstrained	min $f(\boldsymbol{x})$
Constrained	min $f(\boldsymbol{x})$ such that $\boldsymbol{g}(\boldsymbol{x})\leq 0$
Goal	min $\boldsymbol{\gamma}$ such that $\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{x}^{\gamma}\leq \boldsymbol{Goal}$
Minmax	min{max $\boldsymbol{f}(\boldsymbol{x})$} such that $\boldsymbol{g}(\boldsymbol{x})\leq 0$
Nonlinear least squares	min $\Sigma\{\boldsymbol{f}(\boldsymbol{x}).*\boldsymbol{f}(\boldsymbol{x})\}$
Nonlinear equations	$\boldsymbol{f}(\boldsymbol{x})=0$
Semi-infinite constrained	min $f(\boldsymbol{x})$ such that $\boldsymbol{g}(\boldsymbol{x})\leq 0$ & $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{v})\leq 0\ \forall \boldsymbol{v}$

* The routines are designed to work with scalars, 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.

2.1. Standard Constrained Optimization (CO)

The standard constrained optimization problem investigated in all the literature available can be described as follows: Find an optimal point $\boldsymbol{x}^*$ which minimizes the function

$$\begin{aligned} &f(\boldsymbol{x})\\ &\text{subject to:}\\ &g_i(\boldsymbol{x})=0,\ i=1, 2, \ldots, m1\\ &g_j(\boldsymbol{x})\leq 0,\ j=m1+1, \ldots, m2\\ &\boldsymbol{x}_l\leq \boldsymbol{x}\leq \boldsymbol{x}_u \end{aligned} \tag{1}$$

where $m1$ and $m2$ are integers, $\boldsymbol{x}$ is an **n**-dimensional vector of unknowns, $\boldsymbol{x}=(x_1, x_2, \ldots, x_n)$, and f, g_i, $i=1, 2, \ldots, m1$ and g_j, $j=m1+1, \ldots, m2$, are real-valued functions of the variables $(x_1, x_2, \ldots, x_n)$. $\boldsymbol{x}_l=(x_{l1}, x_{l2}, \ldots, x_{ln})$ and $\boldsymbol{x}_u=(x_{u1}, x_{u2}, \ldots, x_{un})$ are the lower and upper bounds of $\boldsymbol{x}$, *respectively*. *The function f* is the objective function, and the equations and inequalities of (1) are constraints.

2.2. Semi-infinite constrained optimization (SICO)

The semi-infinite constrained optimization 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:

$$\begin{aligned} &\text{Minimize}\\ &f(\boldsymbol{x}), \boldsymbol{x}\in \Re^n,\\ &\text{subject to:}\\ &g_i(\boldsymbol{x})=0,\ i=1, 2, \ldots, m1\\ &g_j(x)\leq 0,\ j=m1+1, \ldots, m2\\ &\Phi_k(\boldsymbol{x}, \boldsymbol{v})\leq 0,\ k=1, 2, \ldots, n\\ &\boldsymbol{x}_l<\boldsymbol{x}<\boldsymbol{x}_u \text{ for all } \boldsymbol{v}\in \Re^2 \end{aligned} \tag{2}$$

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

The difference between the CO and SICO is obvious. The latter allows an extra set of independent variables $\boldsymbol{v}$ 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 can not be converted into the former. For more details, see references [32, 44, 45].

The procedure for solving such a SICO with nonlinear constraints is as follows:

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

3. OBSTACLES AND ROBOT REPRESENTATION IN 3D SPACE

Clearly the first thing for robot path planning is to give each of the objects including obstacles and robot in the workspace a mathematical representation. Modeling and manipulation of objects is the research task of Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), and Computer Graphics (CG).[46–53] A solid model should contain an informationally complete description of the geometry and topology of a 3D object.[49] A successful modeling system, in addition to many other 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 modeling systems, namely boundary representation (B-rep), spatial decomposition, and constructive solid geometry (CSG). Different modeling approaches have their own advantages and disadvantages. Chiyokura gives detailed descriptions, comparisons, advantages, and disadvantages of these and other modeling approaches.[48]

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.[1,46,47,51–52] 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,

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

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. Therefore this approach is good for modeling typical subsea structures (manifolds, platforms, pipelines, wellheads, benthic stations). It is, moreover, a convenient representation used by CAD/CAM modeling 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.[1,47,53]

3.1. 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

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

$$\boldsymbol{I} \cap \boldsymbol{B} = \boldsymbol{B} \cap \boldsymbol{T} = \boldsymbol{I} \cap \boldsymbol{T} = \boldsymbol{\phi} \tag{4}$$

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 $\boldsymbol{I}$, $f(p)=I$ 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 $\boldsymbol{T}^c=1$. 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

$$f(p)=1 \tag{5}$$

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

$$f(p)=(x/R)^2+(y/R)^2+(z/R)^2 \tag{6}$$

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

3.2. Boolean operations (intersection and union)

One major goal of a solid modeling 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

$$f^I(p)=\max(f_1(p), f_2(p), \ldots, f_n(p)) \tag{7}$$

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

$$\max(f_1(p), f_2(p), \ldots, f_n(p))=1 \tag{8}$$

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

$$f_1(p)=(x/r)^2, f_2(p)=(y/r)^2, f_3(p)=(z/r)^2 \tag{9}$$

has the following surface equation

$$\max((x/r)^2, (y/r)^2, (z/r)^2)=1 \tag{10}$$

which represents the surface of a cube centered 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

$$f^U(p)=\min(f_1(p), f_2(p), \ldots, f_n(p)) \tag{11}$$

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

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

3.3. Smooth approximation of intersection and union operations

Although Equations (8) and (12) represent the exact surfaces of the intersection and union of the n objects, they are not readily manipulated and computed. To realize a smooth blending of the n objects into a final one, Equations (8) and (12) 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[53]. We use here that method. The intersection and union can be smoothly approximated as:

$$f^I(p)=(f_1^m+f_2^m+\ldots+f_n^m)^{1/m} \tag{13}$$

$$f^U(p)=(f_1^{-m}+f_2^{-m}+\ldots+f_n^{-m})^{-1/m} \tag{14}$$

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[53] proved that when $m\rightarrow\infty$, the approximations (13) and (14) give the exact description of the intersection and union respectively. These approximations have the following advantages:

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

(i) Blending effects are primarily noticeable near surface intersections.
(ii) $f^I(p)$ 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:

$$(f_1^m + f_2^m + \ldots + f_n^m)^{1/m} = 1 \tag{15}$$

$$(f_1^{-m} + f_2^{-m} + \ldots + f_n^{-m})^{(-1/m)} = 1 \tag{16}$$

3.4. Useful primitive solids for obstacle representation and the choice of m

There are many useful defining functions for obstacle representation. The most useful ones include:[47]

a. Quadrics. A frequently used class of objects are the quadric surfaces, which are described with second-degree equations (quadratics). They include spheres, ellipsoids, tori, paraboloids, and hyperboloids. Quadric surfaces, particularly spheres and ellipsoids, are common elements of CAD and Graphics, and are often available in CAD and graphics packages as primitives from which more complex objects can be constructed.

Sphere: In Cartesian coordinates, a spherical surface with radius r centered on the coordinate origin is defined as the points (x, y, z) that satisfy the equation

$$x^2 + y^2 + z^2 = r^2 \tag{17}$$

Ellipsoid: An ellipsoidal surface can be described as an extension of a spherical surface, where the radii in the three mutually perpendicular directions can have different values. The Cartesian representation for points over the surface of an ellipsoid centered on the origin is

$$(x/r_x)^2 + (y/r_y)^2 + (z/r_z)^2 = 1 \tag{18}$$

Slabs, i.e. region bounded by two parallel planes with the expression of $(x/a)^2 = 1$, $(y/b)^2 = 1$, $(z/c)^2 = 1$ and circular or elliptical cylinder with the expression of $(x/a)^2 + (y/h)^2 = 1$ are special cases of this general ellipsoid, here a, b, c are positive real number.

Torus: A torus is a doughnut-shaped object. It can be generated by rotating a circle or other conic about a specified axis. The Cartesian representation for points over the surface of a torus can be written in the form

$$[r - \sqrt{(x/r_x)^2 + (y/r_y)^2}]^2 + (z/r_z)^2 = 1 \tag{19}$$

where r is any given offset value.

b. Superquadrics. This class of objects is a generalization of the quadric representations. It was first developed by Barr[51] and provides more flexibility to describe objects. Superquadrics are formed by incorporating additional parameters into the quadric equations to provide increased flexibility for adjusting object shapes. The number of additional parameters used is equal to the dimension of the object: one parameter for curves and two parameters for surfaces. The most useful one for CSG is superellipsoid.

Superellipsoid. A Cartesian representation for points over the surface of a superellipsoid is obtained from the equation for an ellipsoid by incorporating two exponent parameters:

$$[(x/r_x)^{2/s_2} + (y/r_y)^{2/s_2}]^{s_2/s_1} + (z/r_z)^{1/s_1} = 1 \tag{20}$$

where parameters s_1 and s_2 can be assigned any positive real value. For $s_1 = s_2 = 1$, we have an ordinary ellipsoid. Superellipsoid is also used to represent the robot in 3D space. We will describe this in more detail in the next section.

c. Blobby functions. Blobby function has been used in computer graphics for representing molecular structure, water droplets and other liquid effects, melting objects, and muscle shapes in the human body. In robotics, it is also useful for describing obstacles. Several models have been developed for representing blobby objects as distribution functions over a region of space. One way to do this is to model objects as combinations of Gaussion density functions, or "bumps". A surface function is then defined as

$$\sum_k b_k e^{-a_k r_k^2} = T \tag{21}$$

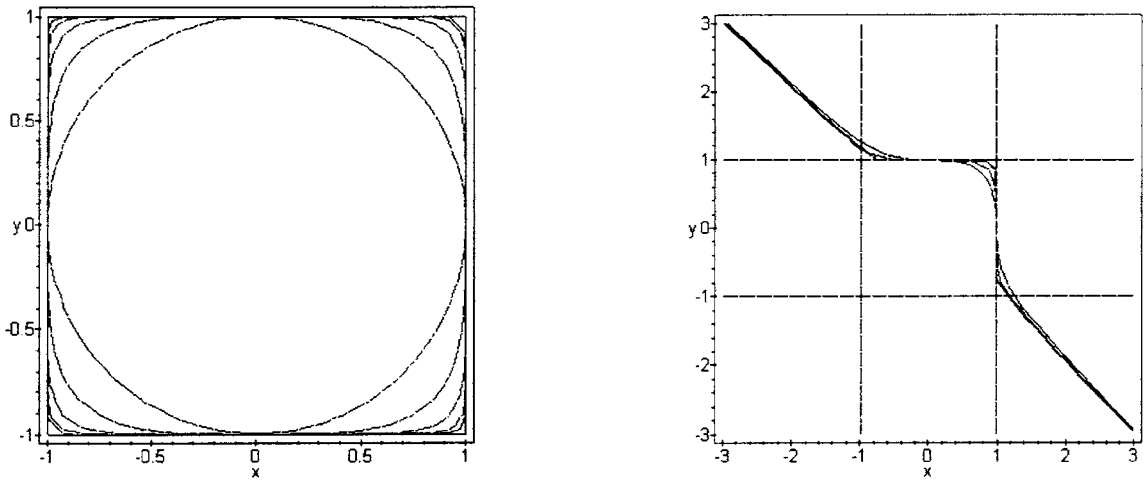
where $r_k^2 = \sqrt{x_k^2 + y_k^2 + z_k^2}$, parameter T is a specified threshold, and parameters a_k and b_k are used to adjust the amount of blobbiness of the individual objects. Negative values for parameter b_k can be used to produce dents instead of bumps.

One problem that has been investigated in the previous literature[1] is the choice of the control parameter m in Equations (15) and (16). For the completeness of the paper it is still included here. Although Ricci[53] 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 2(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 3(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 2(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. Only in the first quadrant it is a good approximation of the intersection of f_1 and f_2. 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.[51] 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 able to keep the closeness of the intersection and union

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press



(a) (b)

Fig. 2. 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.

operation. Figures 3 and 4 give two examples. Similarly as m increases, the approximation gets better.

3.5. Translation and rotation

The defining functions (17) to (21) 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 $[(x-a)/a]^2=1$, which describes the surface of an infinite slab centered at $\underline{x}=a$ and with the same thickness of $2a$. More generally, let $\boldsymbol{M}\in\boldsymbol{R}^{3\times3}$ 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:

$$\underline{\boldsymbol{x}}=\boldsymbol{M}\boldsymbol{x}+\boldsymbol{B} \tag{22}$$

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

$$\underline{f}(\underline{x}, \underline{y}, \underline{z})=f(x, y, z) \tag{23}$$

where

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}=M^{-1}\begin{pmatrix} \underline{x}-b_1 \\ \underline{y}-b_2 \\ \underline{z}-b_3 \end{pmatrix} \tag{24}$$

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

In summary, we can conclude from the presentation given in Sections 3.1 to 3.5 that the boundary, inside and outside of an obstacle can be described as implicit functions in the following general way:

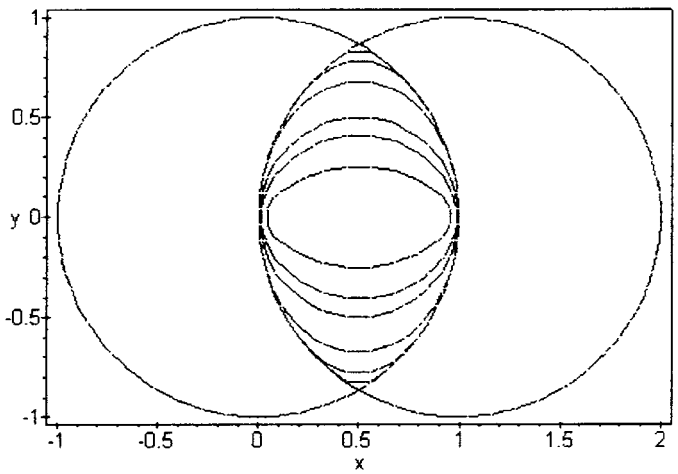


Fig. 3. Illustration of Boolean operation: Intersection $f_1 \cap f_2$. $f_1=x^2+y^2$, $f_2=(x-1)^2+y^2$. $m=0.6$, 0.8, 1, 2, 5, and 25 from inside to outside.

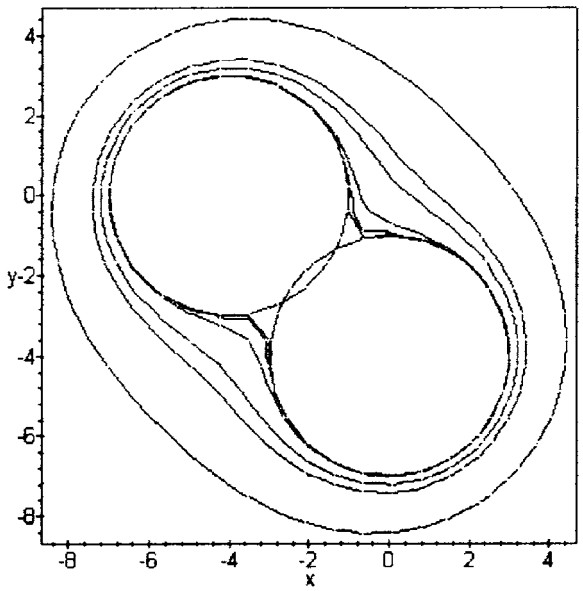


Fig. 4. Illustration of Boolean operation: Union $f_1 \cup f_2$. $f_1=[(x+4)^2+y^2]/3^2$, $f_2=[x^2+(y+4)^2]/3^2$. $m=8$, 4, 2, 1, 0.8, *and* 0.5 *from inside to outside*.

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

$$h(x, y, z)=0 \quad (25)$$

$$h(x, y, z)<0 \quad (26)$$

$$h(x, y, z)>0 \quad (27)$$

3.6. Robot representation

To clearly model and manipulate a robot in a 3D space, we must be capable of representing its spatial location and orientation, its size and shape. To this end, it is well known that six independent variables must be used to represent its spatial position and orientation. Normally a point O, denoted by $\boldsymbol{O}(x_0, y_o, z_0)$, is chosen at the geometric centre of the robot as three independent position variables, and three orientation angles, denoted by $\theta(\theta_1, \theta_2, \theta_3)$, (for example, a set of three Euler angles) are chosen as the other three independent variables.

There are two ways to describe a robot moving in a 3D space. The first is the implicit function which takes the form of $g(x, y, z)=0$ for its boundary expression.[46,52] An explicit function $z=z(x, y)$ is a special case of the general implicit function $g(x, y, z)=0$ due to the fact that $g(x, y, z)=z-z(x, y)$. The second is the parametric form.[46,51] The x, y and z are expressed as functions of two auxiliary parameters $\boldsymbol{v}(t_1, t_2)$, so that $x=x(\boldsymbol{v})=x(t_1, t_2)$, $y=y(\boldsymbol{v})=y(t_1, t_2)$, and $z=z(\boldsymbol{v})=z(t_1, t_2)$. In the following, we use both the implicit and the parametric forms to formulate a robot.

A generalised implicit function representation is given as [28]:

$$g(x, y, z, x_o, y_o, z_o, \theta_1, \theta_2, \theta_3)=0 \quad (28)$$

and its equivalent parametric form can be expressed as [29]:

$$\begin{aligned} x&=x(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}) \\ y&=y(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}) \\ z&=z(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}) \end{aligned} \quad (29)$$

where x, y, z are boundary points of a robot and mainly responsible for the robot's collision free condition. x_o, y_o, z_o define the robot's spatial location, and θ_1, θ_2, θ_3 are orientation angles to represent the robot's spatial orientation. x_o, y_o, z_o together with θ_1, θ_2, θ_3 are responsible for determining the position and orientation of a robot and are called configuration variables. $\boldsymbol{v}$ is a vector with two parameters (t_1, t_2) to specify the robot's boundary points.

The function we use to represent robot concretely is the superellipsoid proposed in references[26,30] which is also a CSG primitive for a broad family of robot and obstacles. The superellipsoid is defined by the function

$$[((x-x_o)/r_x)^{2/s2}+((y-y_o)/r_y))^{2/s2}]^{s2/s1}+((z-z_o)/r_z)^{2/s1}=1 \quad (30)$$

the parametric form is:

$$\begin{aligned} x&=r_x \cos^{s1}(t_1 \cos^{s2}(t_2) \\ y&=r_y \cos^{s1}(t_1) \sin^{s2}(t_2) \quad \text{where} \quad -\pi/2 \leqslant t_1 \leqslant \pi/2 \end{aligned} \quad (31)$$

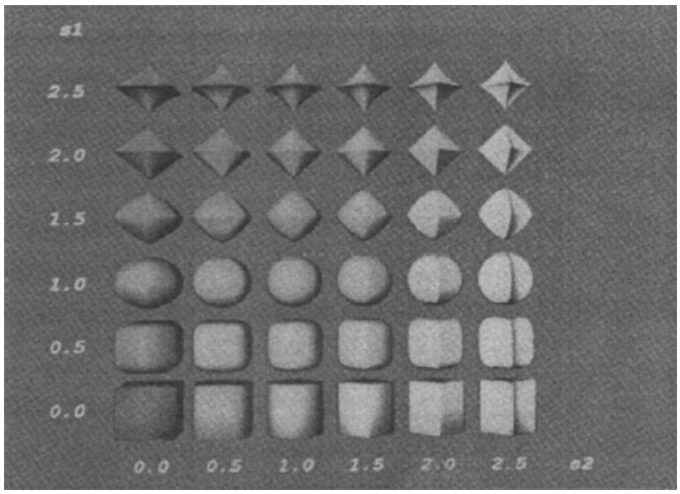


Fig. 5. Superellipsoids plotted with different values for parameters s_1 and s_2 when $r_x=r_y=r_z$.

$$z=r_z \operatorname{sibn}^{s1}(t_1) \qquad 0 \leqslant t_2 \leqslant 2\pi$$

where r_x, r_y, r_z define the geometric extents, s_1 and s_2 specify the shape properties, and x_o, y_o, z_o describe the spatial location, respectively. s_1 is the squareness parameter in the north-south direction; s_2 is the squareness parameter in the east-west direction, t_1 and t_2 are two auxiliary parameters. The superellipsoid can be constructed from the basic slabs. Some superellipsoid shapes produced by the choice of different values for parameters s_1 and s_2 are shown in Figure 5 when $r_x=r_y=r_z$. When $s_1=s_2=1$, the superellipsoid becomes an ordinary ellipsoid that is suitable to describe a path planning vehicle such as an autonomous submarine. When $s_1=s_2=0.5$, the superellipsoid becomes a rectangle with round angle that is suitable for describing a traditional underwater vehicle; while when $s_1=s_2=2$, the superellipsoid becomes a diamond that is suitable for describing a ship. From Berger,[47] we know that most kinds of robots can be simulated by the broad family of easily defined superellipsoid primitives. In addition to the supersphere shapes that can be generated using various values for parameters s_1 and s_2, other superquadratic shapes can also be combined to create more complex structures. More details about them can be found in the references [46, 47].

Equation (30) describes a 3D robot in the standard orientation with $\theta_1=\theta_2=\theta_3=0$. It is necessary to give a general representation of its spatial orientation. The concepts of Euler angle and Euler angle conversion are introduced in the following.

The Euler angles (α, β, γ) comprise three arbitrary rotations in 3D space to describe the spatial orientation of an object. How the Euler angle is defined and how the rotation matrix $\boldsymbol{R}$ is obtained is briefly described, with the assumption that we start in frame S with cartesian axes, x_{old}, y_{old} and z_{old}. A positive (anti-clockwise) rotation of the magnitude α about the z axis of S is first carried out and the resulting frame is called S′. Then it is followed by a positive rotation of the magnitude β about the y' axis of frame S′ and the resulting frame is called S″. Finally, a positive rotation of the magnitude γ about the z'' axis of S″ is made and the resulting frame is called S‴. Figure 6 illustrates the combined effect of these steps.

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

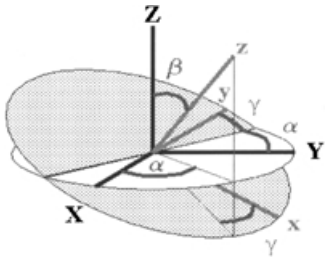


Fig. 6. Euler angle conversion.

The combined result of these three rotations is mathematically expressed by the following rotation matrix:

$$\begin{pmatrix} x_{new} \\ y_{new} \\ z_{new} \end{pmatrix} = R \begin{pmatrix} x_{old} \\ y_{old} \\ z_{old} \end{pmatrix} \tag{32}$$

where $\boldsymbol{R}$ is the rotation matrix:

$$R = \begin{pmatrix} cos\ \alpha \cos\beta \cos\gamma - \sin\alpha \sin\gamma & \sin\alpha \cos\beta \cos\gamma + \cos\alpha \sin\gamma & -\sin\beta \cos\gamma \\ -\cos\alpha \cos\beta \sin\gamma - \sin\alpha \cos\gamma & \sin\alpha \cos\beta \sin\gamma - \cos\alpha \cos\gamma & \sin\beta \sin\gamma \\ \cos\alpha \sin\beta & \sin\alpha \sin\beta & \cos\beta \end{pmatrix}$$

Equation (32) is equivalent to Equation (33):

$$\begin{aligned} x_{new} &= x_{old} \cdot (\cos\alpha \cos\beta \cos\gamma - \sin\alpha \sin\gamma) + y_{old} \\ &\quad \cdot (\sin\alpha \cos\beta \cos\gamma + \cos\alpha \sin\gamma) - z_{old} \cdot (\sin\beta \cos\gamma) \\ y_{new} &= x_{old} \cdot (-\cos\alpha \cos\beta \sin\gamma - \sin\alpha \cos\gamma) + y_{old} \\ &\quad \cdot (\sin\alpha \cos\beta \sin\gamma - \cos\alpha \cos\gamma) + z_{old} \cdot (\sin\beta \sin\gamma) \\ z_{new} &= x_{old} \cdot (\cos\alpha \sin\beta) + y_{old} \cdot (\sin\alpha \sin\beta) + z_{old} \cdot \cos\beta \end{aligned} \tag{33}$$

where vector $(x_{old}, y_{old}, z_{old})$ represent the point in the first coordinate system and $(x_{new}, y_{new}, z_{new})$ represent the same point in the new coordinate system. The ranges for α, β. γ are

$$0 \leqslant \alpha \leqslant 2\pi,\ 0 \leqslant \beta \leqslant \pi,\ 0 \leqslant \gamma \leqslant 2\pi \tag{34}$$

The combination of rotation transformation (33) with Equation (31) results in the parametric form for a super-ellipsoid robot in a general position $\boldsymbol{O}(x_o, y_o, z_o)$ and orientation $\theta(\theta_1, \theta_2, \theta_3)$:

$$\begin{aligned} x &= (r_x \cos^{s1}(t_1) \cos^{s2}(t_2)) l_1 + (r_y \cos^{s1}(t_1) \sin^{s2}(t_2)) l_2 \\ &\quad + (r_z \sin^{s1}(t_1)) l_3 + x_o \\ y &= (r_x \cos^{s1}(t_1) \cos^{s2}(t_2)) m_1 + (r_y \cos^{s1}(t_1) \sin^{s2}(t_2)) m_2 \\ &\quad + (r_z \sin^{s1}(t_1)) m_3 + y_o \\ z &= (r_x \cos^{s1}(t_1) \cos^{s2}(t_2)) n_1 + (r_y \cos^{s1}(t_1) \sin^{s2}(t_2)) n_2 \\ &\quad + (r_z \sin^{s1}(t_1)) n_3 + z_o \end{aligned} \tag{35}$$

where

$$\begin{aligned} l_1 &= \cos(\theta_1) \cos(\theta_2) \cos(\theta_3) - \sin(\theta_1) \sin(\theta_3) \\ m_1 &= -\sin(\theta_1) \cos(\theta_3) - \cos(\theta_1) \cos(\theta_2) \sin(\theta_3) \\ n_1 &= \cos(\theta_1) \sin(\theta_2) \\ l_2 &= -\cos(\theta_1) \sin(\theta_3) + \sin(\theta_1) \cos(\theta_2) \sin(\theta_3) \\ n_2 &= \sin(\theta_2) \sin(\theta_3) \\ l_3 &= \cos(\theta_1) \sin(\theta_2) \\ m_3 &= \sin(\theta_1) \sin(\theta_2) \\ n_3 &= \cos(\theta_2) \\ -\pi/2 &\leqslant t_1 \leqslant \pi/2 \\ 0 &\leqslant t_2 \leqslant 2\pi \end{aligned}$$

where the oriental angles θ_1, θ_2, θ_3 are specified by three Euler angles. For more details about Euler angle conversion, see reference.[54]

4. CONVERTING THE ROBOT PATH PLANNING PROBLEM INTO THE SEMI-INFINITE CONSTRAINED OPTIMIZATION

4.1. The necessary and sufficient condition for a collision free robot in three-dimensional space

The position of a point in 3D space can be described by three variables: (x, y, z). The boundary of an obstacle in 3D space can be described as:

$$h_1(x, y, z) = 1 \tag{36}$$

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

$$1 - h_1(x, y, z) \leq 0 \tag{37}$$

Expressing the free space as the form of inequality constraint (37) is necessary to make it consistent with the requirements of MATLAB's optimization toolbox.[55] The necessary and sufficient condition for a point which is collision-free with this obstacle is that it must fall into the free space. Mathematically, this condition is equivalent to the satisfaction of inequality (37).

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

$$h_j(x, y, z) = 1, \qquad j = 1, 2, \ldots, n \tag{38}$$

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

$$1 - h_j(x, y, z) \leq 0, \qquad j = 1, 2, \ldots, n \tag{39}$$

To make sure a robot is collision free, a necessary and sufficient condition is that all the points on the robot's surface must be collision free. We can get a generalized form of necessary and sufficient condition by substituting (29) into inequalities (39):

$$\begin{aligned} 1 - h_j(&x(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}), y(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}), \\ &z(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v})) \leq 0 \end{aligned} \tag{40}$$

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

In path planning the configuration variables should also meet the limits of the workspace. This requirement can be described as follows:

$$x_{ol} \le x_o \le x_{ou},\ y_{ol} \le y_o \le y_{ou},\ z_{ol} \le z_o \le z_{ou},\ \theta_{1l} \le \theta_1 \le \theta_{1u},$$
$$\theta_{2l} \le \theta_2 \le \theta_{2u},\ \theta_{3l} \le \theta_3 \le \theta_{3u} \tag{41}$$

where (x_{ol}, y_{ol}, z_{ol}) and (x_{ou}, y_{ou}, z_{ou}) are the lower and upper bounds respectively. Equations (40) and (41) form the inequality constraints required in the formulation of the non-linear programming problem for path planning.

When the robot is represented by a superellipsoid, we can get a more specific form of necessary and sufficient condition by substituting (35) into inequalities (40):

$$\begin{aligned}
&1 - h_j((r_x \cos^{s1}(t_1) \cos^{s2}(t_2)) \\
&\cdot(\cos(\theta_1)\cos(\theta_2)\cos(\theta_3)) - \sin(\theta_1)\sin(\theta_3)) - (r_y \cos^{s1}(t_1) \sin^{s2}(t_2)) \\
&\cdot(\cos(\theta_1)\sin(\theta_3) - \cos(\theta_2)\sin(\theta_2)\sin(\theta_3)) + (r_z \sin^{s1}(t_1)) \\
&\cdot(\cos(\theta_1)\sin(\theta_2)) + x_o,\ -(r_x \cos^{s1}(t_1)\cos^{s2}(t_2)) \\
&\cdot(\sin(\theta_1)\cos(\theta_3) + \cos(\theta_1)\cos(\theta_2)\sin(\theta_3)) - (r_y \cos^{s1}(t_1)\sin^{s2}(t_2)) \\
&\cdot(\cos(\theta_1)\cos(\theta_3) + \sin(\theta_1)\cos(\theta_2)\sin(\theta_3)) + (r_z \sin^{s1}(t_1)) \\
&\cdot(\sin(\theta_1)\sin(\theta_2)) + y_o,\ (r_x \cos^{s1}(t_1)\cos^{s2}(t_2)) \\
&\cdot(\cos(\theta_1)\sin(\theta_2)) + (r_y \cos^{s1}(t_1)\sin^{s2}(t_2)) \\
&\cdot(\sin(\theta_2)\sin(\theta_3)) + (r_z \sin^{s1}(t_1)) \\
&\cdot\cos(\theta_2) + z_o) \le 0
\end{aligned} \tag{42}$$

where

$$-\pi/2 \le t_1 \le \pi/2,\ 0 \le t_2 \le 2\pi$$

In Equation (42), s_1 and s_2 are constants for a particular superellipsoid. It's easy to observe from the comparison of (42) with (2) that (42) is the standard form of constraints in the semi-infinite constrained optimization, with $\boldsymbol{v} = (t_1, t_2)$ and $\boldsymbol{x} = (\boldsymbol{O}, \boldsymbol{\theta}) = (x_o, y_o, z_o, \theta_1, \theta_2, \theta_3)$.

4.2. 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 use a quadratic function of form (43) as the objective function, with the goal configuration point (x_{og}, y_{og}, z_{og}) and goal configuration angles $(\theta_{1g}, \theta_{2g}, \theta_{3g})$ being its unique global minimum point and satisfying the condition that $\min f(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3) = f(x_{og}, y_{og}, z_{og}, \theta_{1g}, \theta_{2g}, \theta_{3g}) = 0$.

$$\begin{aligned}
f(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3) = w((x_o - x_{og})^2 + (y_o - y_{og})^2 + (z_o - z_{og})^2) \\
+ (1 - w)((\theta_1 - \theta_{1g})^2 \\
+ (\theta_2 - \theta_{2g})^2 + (\theta_3 - \theta_{3g})^2)
\end{aligned} \tag{43}$$

where w is a non-negative weighted factor that satisfies: $0 \leqslant w \leqslant 1$.

In Equation (43), w is the weight to adjust the relative effects of the spatial position $(x_o - x_{og})^2 + (y_o - y_{og})^2 + (z_o - z_{og})^2$ and the spatial orientation $(\theta_1 - \theta_{1g})^2 + (\theta_2 - \theta_{2g})^2 + (\theta_3 - \theta_{3g})^2$. When $w = 1$, only the effect of the spatial position factor is considered. When $w = 0.5$, both factors are considered equally. w can be adaptively adjusted and be revised during the searching process. (42) and (43) together form a semi-infinite constrained optimization problem. If we use the initial configuration variables $(x_{os}, y_{os}, z_{os}, \theta_{1s}, \theta_{2s}, \theta_{3s})$ as the initial estimate of the optimization problem, the optimum search for $(x_o^*, y_o^*, z_o^*, \theta_1^*, \theta_2^*, \theta_3^*)$ is equivalent to searching the goal configuration variables. If the algorithm is convergent and the problem has a solution, then we will find that $x_o^* = x_{og}$, $y_o^* = y_{og}$, $z_o^* = z_{og}$, $\theta_1^* = \theta_{1g}$, $\theta_2^* = \theta_{2g}$, $\theta_3^* = \theta_{3g}$. Simulation results will be shown later.

In summary, the fundamental idea for this approach is to represent the free space determined by the robot and obstacles as inequality constraints for a semi-infinite constrained optimization problem in three-dimensional space. The goal configuration is designed as the unique global minimum point of the objective function. The initial configuration is treated as the first search point for the optimization problem. Then the numerical algorithm developed for solving the semi-infinite constrained optimization problem can be applied to solve the robot motion planning problem. Every point generated using the semi-infinite constrained optimization method is guaranteed to be in the free space and therefore is collision free.

5. IMPLEMENTATION CONSIDERATIONS AND SIMULATION RESULTS

When implementation is carried out, we only consider the motion of the robot in 3D space because the case for 2D is relatively simple. The vehicle is modeled as a superellipsoid with different shapes and the obstacles are modeled by circle, ellipsoid, cylinder, and various other shapes of superellipsoid.

5.1. Algorithm implementation consideration

The implementation of the semi-infinite constrained optimization is based on the constrained optimization toolbox,[37] but some modifications have been made. The first is the control of the output. In MATLAB,[55] only the final result of the vector $\boldsymbol{x}(x_o^*, y_o^*, z_o^*, \theta_1^*, \theta_2^*, \theta_3^*)$ is provided. However, the important thing for the robot path planning problem is to generate a smooth path. Therefore, a new function which outputs current $(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3)$ at every iteration has been added. The second is the control of the change of Euler Angles in the line search algorithm for every iteration. Recall that the principle of developing an algorithm in

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

nonlinear programming is to minimize the number function evaluations which represents the most efficient way for finding the optimum $\boldsymbol{x}^*$. Thus, the change step is automatically chosen as large as possible to minimize 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, so a modification has been made. For more details on the implementation of the semi-infinite optimization algorithms, see the reference.[55]

5.2. Simulation results

In the following the results obtained for 3D path planning will be given. In all the experiments the algorithm adopted is the Sequential Quadratic Programming (SQP).[40,55] The limits of the workspace are: $\boldsymbol{O}_l=(-20,-20,-20)$ and $\boldsymbol{O}_u=(60,60,60)$, thus the workspace is surrounded by a cube with length 80(–20, 60), width 80(–20, 60), and height 80(–20, 60). The inequality $\boldsymbol{O}_l \leq \boldsymbol{O} \leq \boldsymbol{O}_u$ will guarantee that the generated path must be in the workspace. Let $\boldsymbol{O}_s=(x_{os}, y_{os}, z_{os})$ and $\boldsymbol{\theta}_s=(\theta_{1s}, \theta_{2s}, \theta_{3s})$ denote the initial configuration and $\boldsymbol{O}_g=(x_{og}, y_{og}, z_{og})$, $\boldsymbol{\theta}_g=(\theta_{1g}, \theta_{2g}, \theta_{3g})$ denote the goal configuration. The robot's start and goal configurations are chosen as $(x_{os}, y_{os}, z_{os}, \theta_{1s}, \theta_{2s}, \theta_{3s})=(-20,-20,-20,0,0,0)$ and $(x_{og}, y_{og}, z_{og}, \theta_1, \theta_2, \theta_3)=(50,50,50,0,0,0)$ respectively. The objective function is defined as:

$$f=w((x_o-50)^2+(y_o-50)^2+(z_o-50)^2)+1-w)(\theta_1^2+\theta_2^2+\theta_3^2) \tag{44}$$

where w is chosen as $0 \leqslant w \leqslant 1$. In the four experiments carried out below, the robot is respectively represented by four different superellipsoids whose shapes are shown in Figure 7.

It is well known from computer graphics that a set of 3D objects may be viewed from different angles. However it is hard to find a view angle that could clearly illustrate the collision-free path even though we have generated such a path using the approach. For this reason two or three views of the generated path from different angles are presented in all of the following experiments to assist 3D visualization.

Example 1. results with circular obstacles

A sphere with center located at the point (x_c, y_c, z_c) can be described as:

$$(x-x_c)^2+(y-y_c)^2+(z-z_c)=R^2 \tag{45}$$

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

$$R^2-(x-x_c)^2-(y-y_c)^2-(z-z_c)^2 \leq 0 \tag{46}$$

then it must be collision-free with this circle.

In order to demonstrate the convergence of the problem formulation to the goal configuration, the workspace considered in example 1 contains seven obstacles as shown in Figures 8(a) and (b). The obstacles are represented as spheres with the following boundary expressions:

$$\begin{aligned}
&(x-20)^2+(y-20)^2+(z-20)^2=225\\
&x^2+(y-20)^2+(z-20)^2=25\\
&(x-40)^2+(y-20)^2+(z-20)^2=25\\
&(x-20)^2+y+(z-20)^2=25\\
&(x-20)^2+(y-40)^2+(z-20)^2=25\\
&(x-20)^2+(y-20)^2+z^2=25\\
&(x-20)^2+(y-20)^2+(z-40)^2=25
\end{aligned} \tag{47}$$

The robot is represented by a superellipsoid with $r_x=5$, $r_y=4$, $r_z=3$, $s_1=s_2=1$, which is an ellipsoid.

Figures 8(a) and (b) show the same generated path, the same obstacles and the same robot, but from different view angles using the optimization toolbox in Matlab. It can be observed that the optimization algorithm does converge to the goal and a smooth path has been generated.

Example 2: results with cylindrical obstacles

In order to illustrate the suitability of the approach for different obstacle shapes, we have shown another distributed-obstacle situation as shown in Figures 9(a) and 9(b). The results simulate a real word path planning task

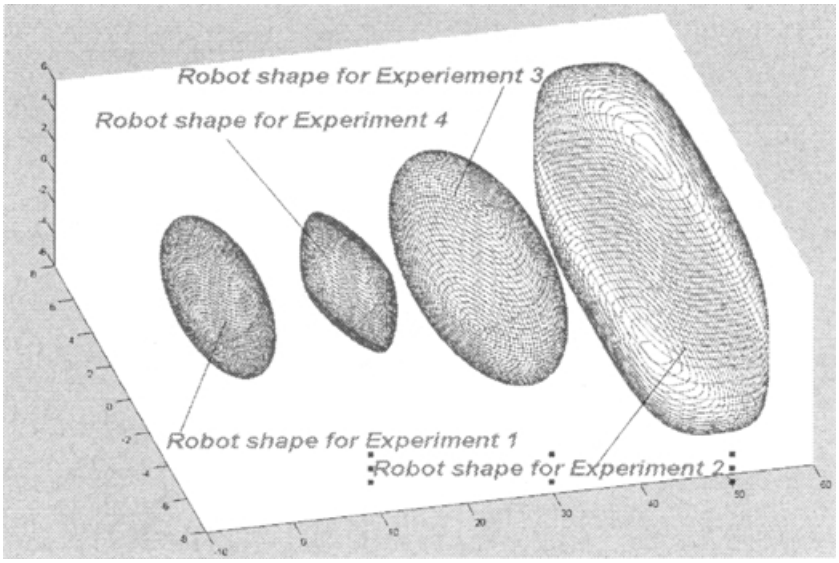
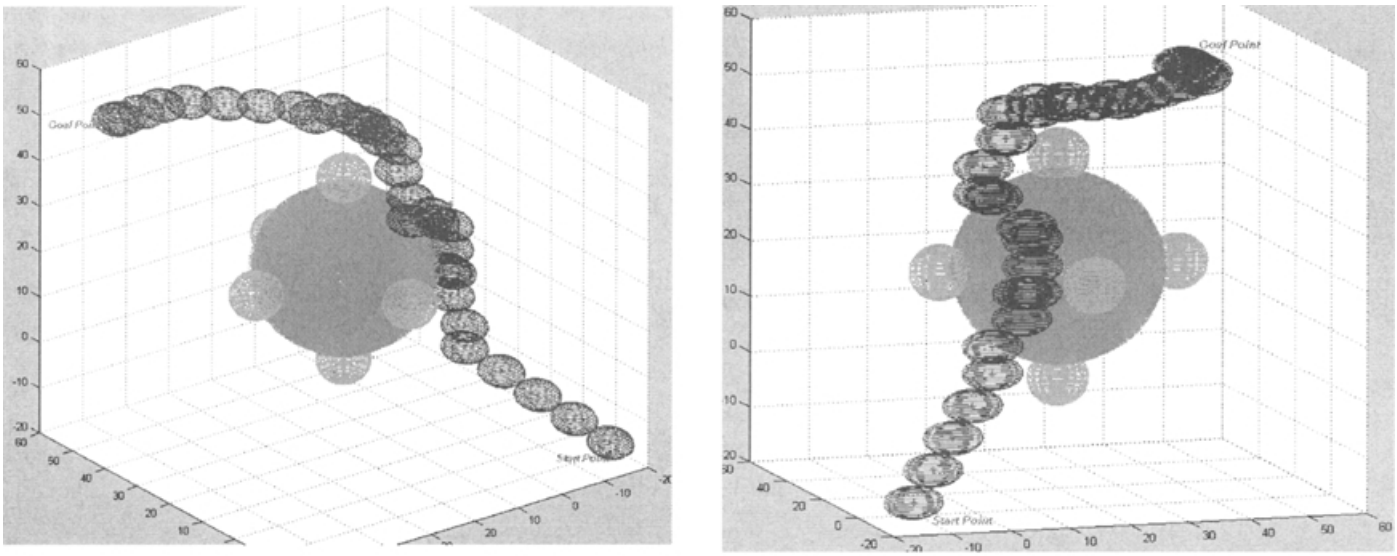


Fig. 7. The four robot shapes used in Examples 1 to 4.

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press



(a) (b)

Fig. 8. (a) Workspace with seven circles and the generated path, viewed from one angle. (b) The same result as that in Fig. 8(a), but viewed from another angle.

encountered in offshore industry. The designed workspace contains a set of cylinders which could be easily recognized as pipelines or a base of an offshore structure. In this example, we have 9 cylinders to represent obstacles with the following boundary equations:

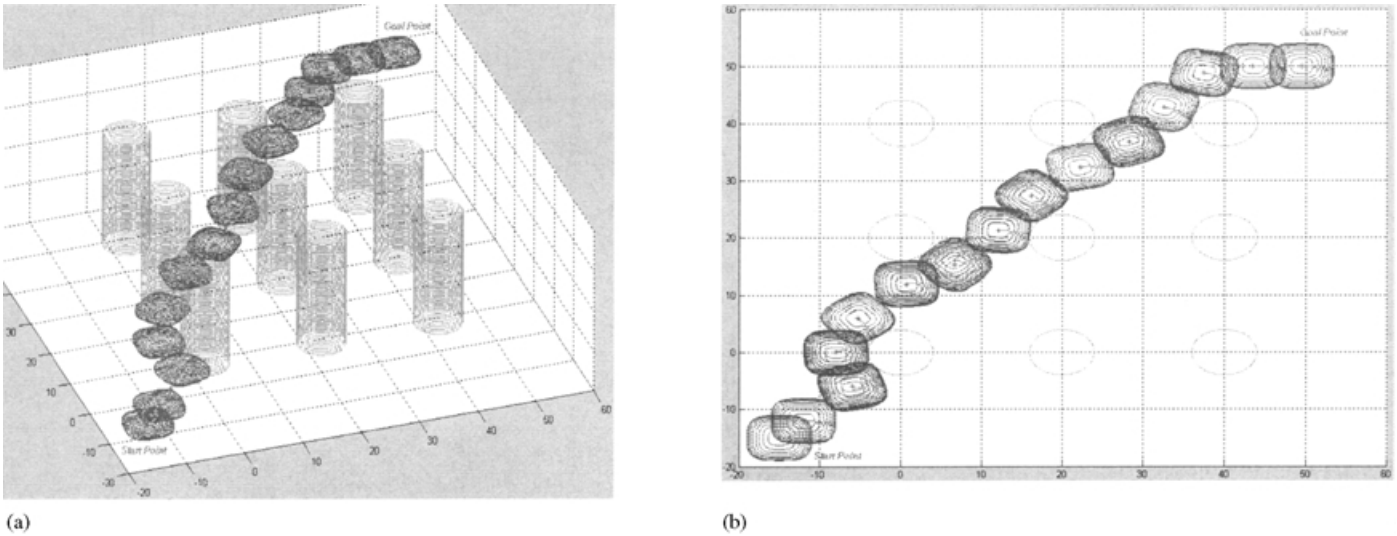
$$\begin{aligned}
&x^2+y^2=16\\
&x^2+(y-20)^2=16\\
&x^2+(y-40)^2=16\\
&(x-20)^2+(y-20)^2=16\\
&(x-20)^2+(y-20)^2=16\\
&(x-20)^2+(y-40)^2=16\\
&(x-40)^2+y^2=16\\
&(x-40)^2+(y-20)^2=16\\
&(x-40)^2+(y-40)^2=16\\
&-20\le z\le 60
\end{aligned} \tag{48}$$

The robot is represented by a superellipsoid with $r_x=5$, $r_y=4$, $r_z=3$, $s_1=s_2=1.5$, which is a general superellipsoid.

Figure 9(a) is a 3D view and Figure 9(b) is a 2D view. In this example the 2D view clearly shows the collision avoidance of the robot from the obstacles. From Figure 9(a) and Figure 9(b), we can also observe that the generated path passes behind the cylinder without touching it and the plotted path avoids all the obstacles and converges to the goal in a smooth way. We have tested many other star points and the results are all quite satisfactory, although they are not reported here for the limitation of space.

Example 3. results with various superellipsoids

To test robustness of the approach to obstacle shapes, Figures 10(a), (b), (c) illustrate simulation results with various superellpsoid obstacles. There are 11 obstacles in the workspace, and among which seven are superellipsoids, two spheres and two ellipsoids. The boundary expressions for them are respectively represented as follows:



(a) (b)

Fig. 9. (a) Workspace with nine cylinders and the generated path, viewed from one angle. (b) A 2D view (xy plane) of the same result as that in Fig. 9(a).

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

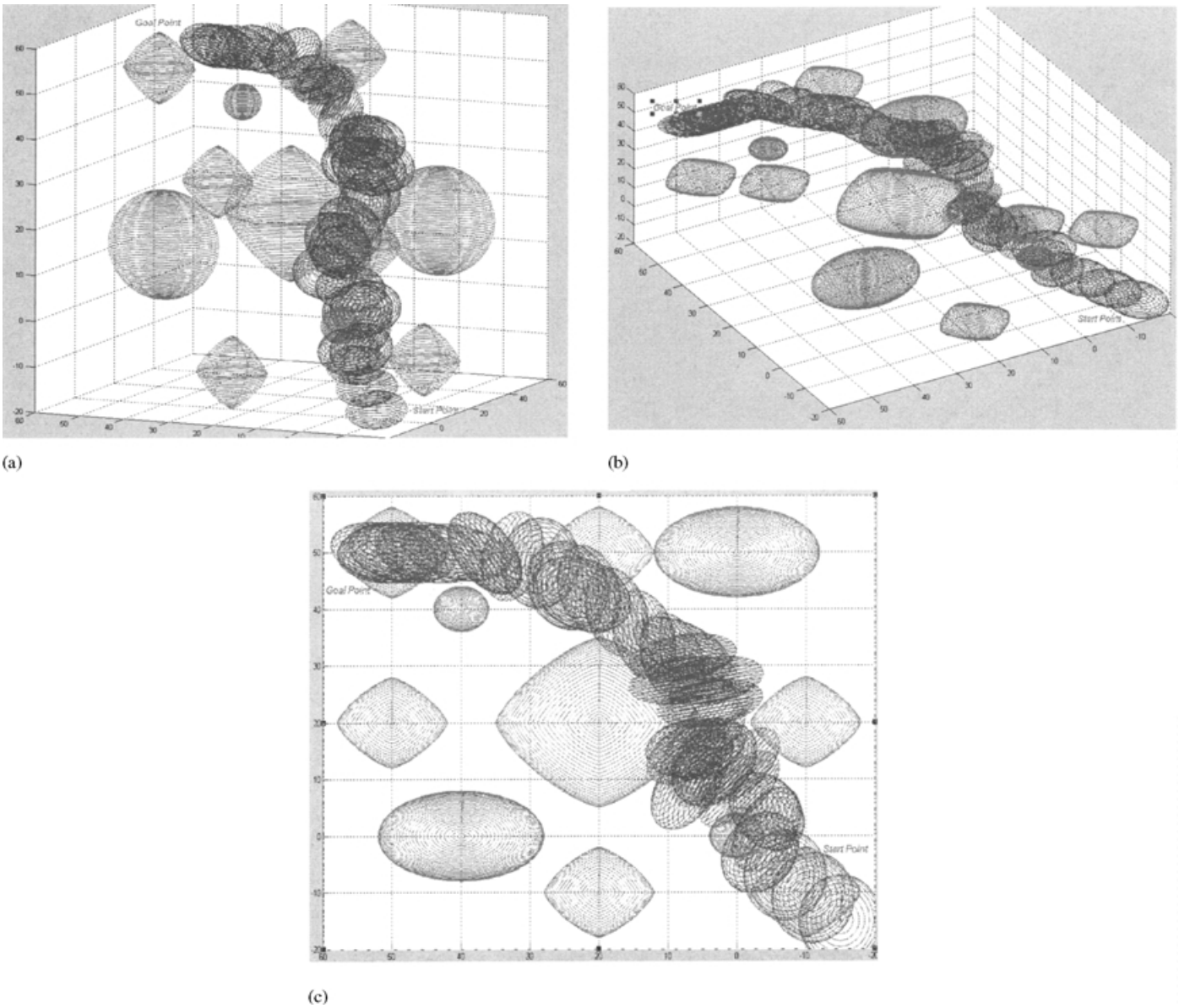


Fig. 10. (a) Simulation result with various superellpsoids from a 3D view angle. (b) Simulation result various superellpsoids from another 3D view angle. (c) Simulation result various superellpsoids from a 2D view angle.

7 superellipsoids:

$$[((x-20)/15)^{4/3}+((y-20)/15)^{4/3}]+((z-20)/15)^{4/3}=1$$
$$[((x-20)/8)^{4/3}+((y+10)/8)^{4/3}+((z+10)/8)^{4/3}=1$$
$$[((x+10)/8)^{4/3}+((y-20)/8)^{4/3}+((z+10)/8)^{4/3}=1$$
$$[((x+10)/8)^{4/3}+((y+10)/8)^{4/3}+((z-20)/8)^{4/3}=1$$
$$[((x-20)/8)^{4/3}+((y-50)/8)^{4/3}+((z-50)/8)^{4/3}=1$$
$$[((x-25)/8)^{4/3}+((y-20)/8)^{4/3}+((z-50)/8)^{4/3}=1$$
$$[((x-50)/8)^{4/3}+((y-50)/8)^{4/3}+((z-20)/8)^{4/3}=1 \quad (49)$$

2 spheres:

$$(x-40)^2+(y-40)^2+(z-40)^2=16$$
$$x^2+y^2+z^2=16 \quad (50)$$

2 ellipsoids:

$$((x-40)/12)^2+(y/8)^2+((z-15)/12)^2=1$$
$$(x/12)^2+((y-50)/8)^2+((z-15)/12)^2=1 \quad (51)$$

The robot is represented by an ellipsoid with $r_x=8$, $r_y=6$, $r_z=4$, $s_1=s_2=1$.

Figure 10(a) and Figure 10(b) are two 3D views while Figure 10(c) is a 2D view. It is obvious that the path generated using the optimization technique avoids the obstacles and converges to the goal. From this example we can observe that this approach has the ability to negotiate very narrow channels among obstacles.

Example 4: results with various superellipsoids and cylinders

In addition to the simulation results shown in Figures 8 to 10, we have carried out another test with mixed superellipsoids and cylinders as shown in Figures 11(a), 11(b) and 11(c). In this test, five obstacles are included and the robot is represented by a superellipsoid with $r_x=8$, $r_y=8$, $r_z=6$, $s_1=s_2=0.8$. The boundary expressions for the obstacles are given by Equations (52) and (53):

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

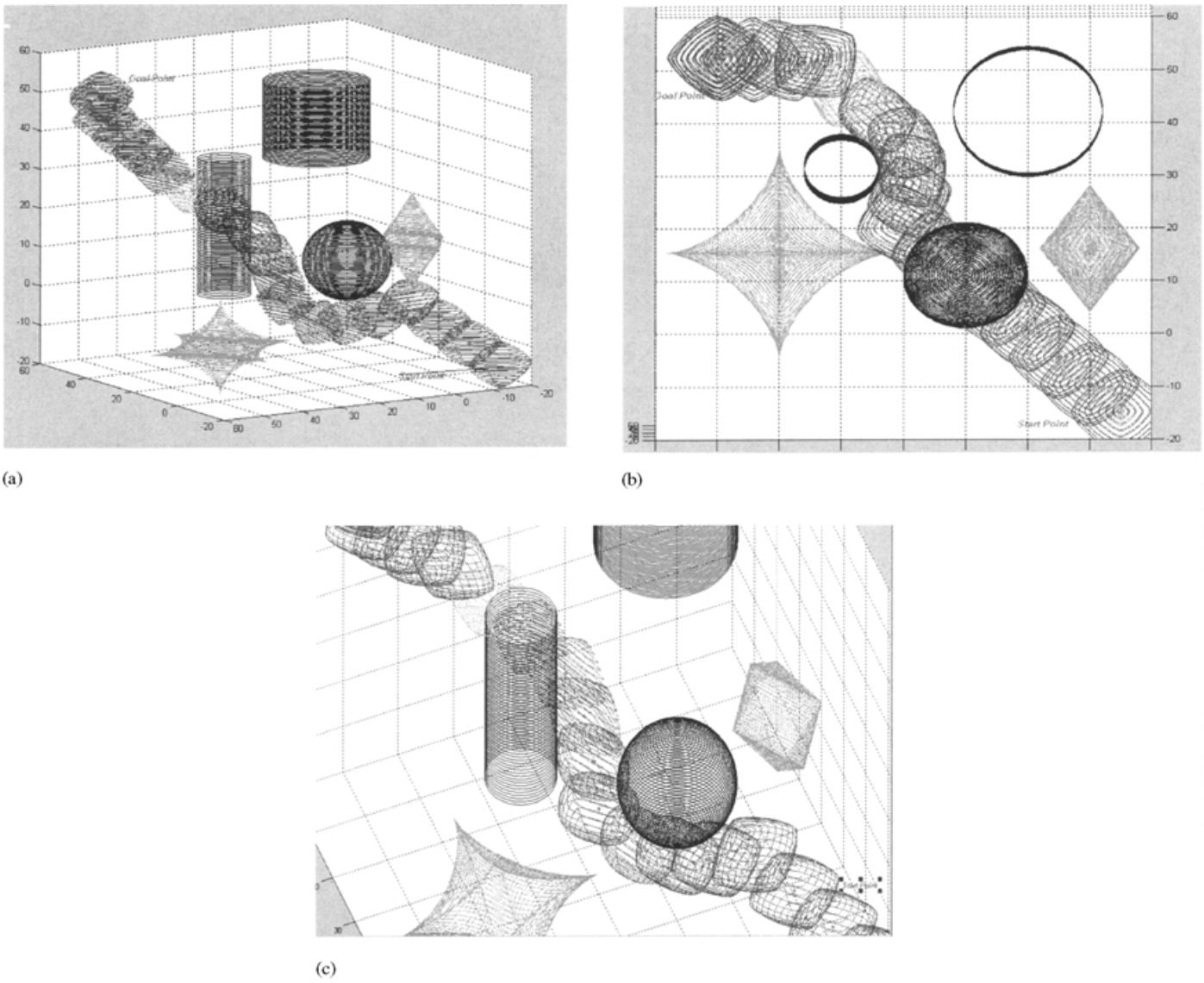


Fig. 11. (a) Simulation result with mixed superellipsoids and cylinders: a 3D view angle. (b) Simulation result with mixed superellipsoids and cylinders: a 2D view. (c) Simulation result with mixed superellipsoids and cylinders: another 3D view.

2 cylinders:

$$(x-40)^2+y^2=144,\ 30<z<50$$
$$(x-30)^2+(y-30)^2=100,\ -5\le z\le 30 \qquad (52)$$

3 superellipoids:

$$((x-15)/12)+((y+10)/8+((z-12)/12)=1\ (s_1=s_2=2\ \textit{here})$$
$$((x-15)/20)^{2/3}+((y-40)/18)^{2/3}+((z+10)/12)^{2/3}=1$$
$$(x-10)^2+(y-10)^2+(z-10)^2=100 \qquad (53)$$

Figure 11(a) is a 3D view and Figure 11(b) is a 2D view, Figure 10(c) is an enlarged view of Figure 11(a). This experiment results show that the robot can adjust its orientation angles autonomously to reach the goal point.

6. CONCLUSION

In this paper, a systematic approach to motion planning for subsea vehicle using the semi-infinite constrained optimization techniques has been presented. We have shown the principle of converting the path planning problem into the standard Semi-infinite Constrained Optimization problem. This direct path planning approach considers the robot's 3D shape and is totally different from the traditional approaches in the way that the calculation of the Cspace obstacles is no longer needed. This new real-time robot path planning approach, to our best knowledge, is so far the first one in the robotic community. In addition, this approach differs 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 suff1cient 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

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

search speed with large numbers of constraints may be overcome by the approximate 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 approach 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. The Semi-infinite Constrained Optimization approach with an adaptive objective function has the following advantages:

(i) The robot does not need to be shrunk to a point and the obstacles do not need to be expended to Cspace, the entire procedure of path planning can be carried out in real 3D space.
(ii) The goal point is guaranteed to be the only global minimum of the objective function.
(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.

The research carried out in this paper has indicated that robot path planning can be formulated as semi-infinite constrained optimization problem. Although the fundamentals for 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. The treatment is consistent. The study presented in this paper has shown its great potential as an on-line motion planner.

Acknowledgement

The authors gratefully acknowledge the financial support from Chinese Academy of Sciences, P. R. China and Royal Society of United Kingdom for the joint research project under grant No. 20030389, 2003-2006, and the financial support from Chinese Academy of Sciences and Chinese State Development Planning Commission for Bai Ren Ji Hua (BRJH), 2002–2005.

References

1. 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).
2. Y. R. Petillot, T. I. Ruiz, D. M. Lane, Y. Wang, E. Trucco and N. Pican, "Underwater vehicle path planning using a multibeam forward looking sonar", *IEEE Oceanic Engineering Society, OCEANS '98*, 1998, Nice, France (1998) pp. 1194–1199.
3. Y. Petillot, I. T. Ruiz and D. M. Lane, "Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar", *IEEE Journal of Oceanic Engineering* **26**, 240–251 (2001).
4. G. Oriolo, G Ulivi and M. Vendittelli, "Real-time map building and Navigation for autonomous robots in unknown environments", *IEEE Trans. on Systems, Man and Cybernetics, Part B: Cybernetics* **28**, 316–333 (1998).
5. H. P. Huang and P. C. Lee, "A real-time algorithm for obstacle avoidance of autonomous mobile robots", *Robotica***10**, Part 2, 217–227 (1992).
6. O. Khatib, "Real-time obstacle avoidance for manipulator and mobile robots", *Int. J. Robotics Research* **5**, 90–98 (1986).
7. C. M. Krishna and K. G Shin, *Real-time systems* (McGraw-Hill, New York, 1997).
8. 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).
9. Yongji Wang, J. A. Linnett and J. Roberts, "Kinematics, kinematic constraints and path planning for wheeled mobile robots", *Robotica* **12**, Part 1, 391–400 (1994).
10. T. J. Park, J. W Ahn and C. S. Han, "A path generation algorithm of an automatic guided vehicle using sensor scanning method", *KSME International Journal* **16**, 137–146 (2002).
11. I. T. Ruiz, D. M. Lane and M. J. Chantler, "A comparison of inter-frame feature measures for robust object classification in sector scan sonar image sequences", *IEEE Journal of Oceanic Engineering* **24**, 458–469 (1999).
12. E. Trucco, Y. R. Petillot, I. T. Ruiz, K. Plakas and D. M. Lane, "Feature tracking in video and sonar subsea sequences with applications", *Computer Vision and Image Uuderstanding* **79**, 92–122 (2000).
13. J. C. Latombe, *Robot Motion Planning* (Kluwer Academic Publishers, Boston, 1991).
14. R. A. Brooks and T. Lozano-Perez, "A subdivision algorithm in configuration space for find-path with rotation", *IEEE Trans. on Systems, Man and Cybernetic* **15**, 224–233 (1985).
15. T. Lozano-Perez, "Spatial planning: A configuration space approach", *IEEE Trans. on Computer* **32**, 108–120 (1983).
16. V. 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).
17. C. I. Connolly, "Harmonic function and collision probabilities", *Int. J. Robotics Research* **16**, 497–507 (1997).
18. Y. K. Hwang and N. Ahuja, "A potential field approach to path planning", *IEEE Trans. on Robotics and Automation* **8**, 23–32 (1992).
19. T. C. Hu, A. B. Kahng and G Robins, "Optimal robust path planning in general environment", *IEEE Trans. on Robotics and Automation* **9**, 755–774 (1993).
20. R. A. Conn and M. Kam, "On the moving-obstacle path planning algorithm of Shih, Lee and Gruver", *IEEE Trans. on Systems, Man and Cybernetics – Part B: Cybernetics* **27**, 136–138 (1997).
21. H. C. Lu and M. E. Yeh, "Robot path planning based on modified grey relational analysis", *Cybernetics and Systems* **33**, 129–159 (2002).
22. J. C. Chang and H. C. Lu, "Backing up a simulated truck via grey relational analysis", *Journal of the Chinese Institute of Engineering* **24**, 745–752 (2001).
23. C. L. Shih and J. T. Jeng, "Stabilization of non-holonomic chained systems by gain scheduling", *International Journal of Systems Science* **30**, 441–449 (1999).
24. S. Sundar and S. Shiller, "Optimal obstacle avoidance based on the HJB equation", *IEEE Trans. on Robotics and Automation* 3, 305–310 (1997).
25. 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: Cybernetics* **27**, 723–724 (1997).
26. Yongji Wang and David M. Lane, "Solving a generalized constrained optimization problem with both logic AND and OR relationships by a mathematical transformation and its application to robot path planning", *IEEE Trans. on Systems, Man and Cybernetics, Part C: Application and Reviews* **30**, 525–536 (2000).

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press

27. Yongji Wang, D. M. Lane and G J. Falconer, "Two novel approaches for unmanned underwater vehicle path planning: constrained optimization and semi-infinite constrained optimization", *Robotica* **18**, 123–142 (2000).
28. Yongji Wang and M. P. Cartmell, "A new overtaking model for Passing Sight Distance (PDS) on two- lane highways", *ASCE J. of Transportation Engineering* **124**, 536–545 (1998).
29. Yongji Wang and M. R Cartmell, "Autonomous vehicle parallel parking design using function fitting approaches", *Robotica* **16**, 159–170 (1998).
30. Yongji Wang and M. P Cartmell, "Trajectory generation for four-wheel steering tractor-trailer system: a two step method", *Robotica* **16**, 381–386 (1998).
31. Yongji Wang and J. A. Linnett, "Vehicle kinematics and its application to highway design", *ASCE J. of Transportation Engineering* **121**, 63–74 (1995).
32. Yongji Wang, J. A. Linnett and J. W. Roberts, "Motion feasibility of a wheeled vehicle with a steering angle limit", *Robotica* **12**, 217–226 (1994).
33. W. L. Xu and B. L. Ma, "Polynomial motion of non-holonomic mechanical systems of chained form", *Mathematical Methods in the Applied Sciences* **22**, 1153–1173 (1999).
34. Y. Zhang' and K. R Valavanis, "A 3-D potential panel method for robot motion planning", *Robotica* **15**, 421–434 (1997).
35. W. R Wahg and K. K. Wang, "Geometric Modeling for swept volume of moving solids", *IEEE CG&A* **12**, 8–17 (1986).
36. K. Tang, M. Wang, L. Chen, S. Chou, T. Woo and R. Janardane, "Computing planar swept polygons under translation", *Computer Aided Design* **29**, 825–836 (1997).
37. D. Blackmore, M. Leu and L. R Wang, "The sweep-envelope differential equation algonthm and its application to NC machining verification", *Computer Aided Design* **29**, 629–637 (1997).
38. E. J. Haug, F. A. Adkins and D. Cororian, "Domains of mobility for planar body moving among obstacles", *Transactions of the ASME, Journal of Mechanical Design* **120**, 462–467 (1988).
39. R. Fletcher, *Practical Methods of Optimization* (John Wiley & Sons., New York, 1987).
40. R E. Gill, W. Murray and M. H. Wright, *Practical Optimization* (Academic Press, London, 1981).
41. S. R Han, "A globally convergent method for nonlinear programming", *J. Optimization Theory and Applications* **22**, 297–235 (1977).
42. D. G Luenberger, *Linear and Nonlinear Programming* (Addison-Wesley Publishing Company, 2nd ed. London, 1984).
43. S. S. Rao, *Optimization Theory and Applications* (Wiley Eastern Ltd, Second Edition, New Delhi, 1984).
44. E. Polak and D. Q. Mayne, "Control system design via semi-infinite optimization: a review", *Proceedings of IEEE* **72**, 1777–1793 (1984).
45. Y. Tanaka, M. Fukushima and T. Ibaraki, "A comparative study of several semi-infinite nonlinear programming algorithms", *European Journal of Operation Research* **36**, 92–100 (1988).
46. A. H. Barr, "Superquadrics and angle-preserving transformations", *IEEE Computer Graphics and Applications* **1**, 11–23 (1981).
47. M. Berger, *Computer Graphics* (The Benjamin/Cummings Publishing Company, Inc., California, 1986).
48. H. Chiyokura, *Solid Modelling with Desigubase* (Addison-Wesley Publishing Limited, New York, 1988).
49. 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.
50. R G Comba, "Aprocedure fordetecting intersections ofthree-dimensional objects", *JACM* **15**, 354–366 (1968).
51. W. R. Franklin and A. H. Barr, "Faster calculation of superquadric shapes", *IEEE Computer Graphics and Application* **1**, 41–57 (1981).
52. M. Hall and J. Warren, "Adaptive polygonalization of implicitly defined surfaces", *IEEE Computer Graphics & Applications* **10**, 33–42 (1990).
53. A. Ricci, "A constructive geometry for computer graphics", *The Computer Journal* **16**, 157–160 (1973).
54. E. M. Rose, *Elementary Theory of Angular Momentum* (John Wiley & Sons, New York, 1957).
55. The Math Works Inc., MATLAB, 1993, *Optimization Toolbox User's Guide.*

https://doi.org/10.1017/S0263574703005472 Published online by Cambridge University Press