We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

20

A Generalized Robot Path Planning Approach Without The Cspace Calculation

Yongji Wang[1], Matthew Cartmell[2] , QingWang[1] , Qiuming Tao[1]
[1]*State Key Laboratory of Computer Science and Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing, China*
[2]*Department of Mechanical Engineering, University of Glasgow, Glasgow, G12 8QQ, U.K.*

1. Introduction

One of the ultimate goals of robotics is to create autonomous robots. Such robots could accept high level instructions and carry out tasks without further human supervision or intervention. High level input commands would specify a desired task and the autonomous robot would compute how to complete the task itself. Progress towards autonomous robots is of great research and practical interest, with possible applications in any environment hostile to humans. Examples are underwater work, space exploration, waste management and bomb disposal among many others. One of the key technical issues towards such an autonomous robot is the Path Planning Problem (PPP): How can a robot decide what paths to follow to achieve its task. The PPP can be described as follows: given a robot with an initial configuration, a goal configuration, its shape and a set of obstacles located in the workspace, find a collision-free path from the initial configuration to the goal configuration for it.

PPP has been an active field during the past thirty years. Although seemingly trivial, it has proved notoriously difficult to find techniques which work efficiently, especially in the presence of multiple obstacles. A significant and varied effort has been made on this complicated problem (Wang et al., 2005; Wang et al., 2004; Wang & Lane, 2000; Wang & Lane, 1997; Wang, 1995; Wang, 1997; Wang et al., 2000; Wang & Cartmell, 1998c; Wang & Cartmell, 1998a; Wang & Cartmell, 1998b; Wang & Linnett, 1995; Wang et al., 1994a; Wang et al., 1994b; Petillot et al., 1998; Petillot et al., 2001; Park et al., 2002; Ruiz et al., 1999; Trucco et al., 2000; Brooks & Lozano-Perez, 1985; Conn & Kam, 1997; Connolly, 1997; Hu et al., 1993; Huang & Lee, 1992; Hwang & Ahuja, 1992; Khatib, 1986; Latombe, 1991; Lozano-Perez, 1983; Lu & Yeh, 2002; Lumelsky, 1991; Oriolo et al., 1998; Xu & Ma, 1999; Zhang & Valavanis, 1997). Various methods for dealing with the basic find-path problem and its extensions, such as Vgraph, Voronoi diagram, exact cell decomposition, approximate cell decomposition, potential field approach, and optimization-based approach have been developed. A systematic discussion on these old methods can be found in references (Wang et al., 2005; Wang, 1995; Latombe, 1991).

In any robot path planning method, robot and obstacle representation is the first thing to be

Source: Mobile Robots: Perception & Navigation, Book edited by: Sascha Kolski, ISBN 3-86611-283-1, pp. 704, February 2007, Plv/ARS, Germany

considered. PPP essentially deals with how to find a collision_free path for a 3 Dimensional (3D) object (robot) moving among another set of 3D objects (obstacles), satisfying various constraints (Wang et al., 2005; Wang et al., 2004; Wang & Lane, 2000; Wang & Lane, 1997; Wang, 1995; Wang, 1997; Wang et al., 2000). There are many reasons for having so many different approaches developed. For example, assumptions made on both the shapes of the robot and obstacles and the constraints imposed by the mechanical structure of the robot contribute to them (Wang, 1997). The important thing for judging the reality of an approach is whether the realistic constraints have been considered.

An important concept proposed in the early stage of robot path planning field is the shrinking of the robot to a point 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 (Lozano-Perez, 1983) in the Artificial Intelligence Laboratory, MIT as a basis of the spatial planning approach for the find-path and find-place problems, and then extended by Latombe (Latombe, 1991) as a basis for all motion planning approaches suitable for a point robot. However, the research experiences obtained so far have shown that the calculation of Cspace obstacles is very hard in 2D when the following situations occur. 1. Both the robot and obstacles are not polygons; and 2. The robot is allowed to rotate. The situation gets even worse when the robot and obstacles are 3D objects with various shapes (Ricci, 1973; Blechschmidt & Nagasuru, 1990; Barr, 1981; Chiyokura, 1988). For this reason, direct path planning approaches without the Cspace calculation is quite useful and expected.

The objective of this chapter is to present a new approach to the PPP without the Cspace calculation. The chapter is based on our previous work (Wang et al., 2005; Wang et al., 2004; Wang & Lane, 2000; Wang & Lane, 1997), and in the following we will present the background of the new method to show its principle.

Historically the Constrained Optimization and Constructive Solid Geometry (COCSG) method is first proposed in (Wang & Lane, 1997), and two assumptions made in it are that: 1. The Cspace obstacles in the workspace can be approximately represented by inequalities; and 2. The robot can be treated as a point. The mathematical foundations for the Constructive Solid Geometry (CSG), the Boolean operations, and the approximation techniques are developed to represent the free space of the robot as a set of inequalities (Ricci, 1973; Wang & Lane, 1997). The fundamental ideas used include: 1. The free Cspace of the robot is represented as a set of inequality constraints using configuration variables; 2. 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; and 3. 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 space, and therefore is collision free. The contribution of the above paper is that for the first time, the idea of inequality is introduced to represent objects and the optimization technique is used for the efficient search.

However, we can still observe that two issues arise from the above problem formulation. One is how to exactly rather than approximately deal with the shapes of both the robot and the obstacles, and the other is how to calculate the Cspace obstacles. In reference (Wang &

Lane, 2000), we further investigate the effect of obstacle shapes on the problem formulation, and introduce the new concept of the first and second kinds of obstacles. When the second kind of obstacles is considered, the PPP leads to a generalized constrained optimization problem (GCOP) with both logic AND and OR relationships, which is totally different from the traditional standard constrained optimization problem with only logic AND relationship among the constraints. A mathematical transformation technique is developed to solve the GCOP. The original contributions of this paper include threefold: First, from the viewpoint of optimization theory, it is the first one to propose such a GCOP; Second, a method is developed to solve such a GCOP; Third, from the viewpoint of PPP, this paper inherits the advantage of the previous method in (Wang & Lane, 1997) and further generalizes its ability to deal with various shapes of obstacles.

The issue that has not been addressed by the above two papers is the calculation of the Cspace obstacles. We deal with the PPP with the first kind of obstacles in (Wang et al., 2004) and the second kind of obstacles in (Wang et al., 2005) respectively, without the need to calculate the Cspace obstacles. 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. The principle is that the points outside the obstacles in the 3D workspace are represented by implicit inequalities, the points on the boundary of a 3D robot are expressed in the form of a parametric function, and the PPP is formulated as a semi-infinite constrained optimization problem with the help of the mathematical transformation. To show its merits, simulation results with different shapes of robot and obstacles in 3D space are presented.

In this chapter we will present a comprehensive introduction to the principle of the PPP without the Cspace calculation, including the mathematical background, robot and obstacle representation, sufficient and necessary condition for collision-free path, algorithm efficiency, and the simulation results. Particularly, we will also discuss the constraints that must be considered in the future work and explain mathematically the reason why these constraints can lead to more difficulties in this area.

The rest of the chapter is organized as follows. Section 2 gives a brief description of inequality constraints and the formulations for optimization theory. In particular, a previously-developed, generalized constrained optimization and the mathematical translation needed for its solution are also presented in this section. In Section 3, obstacle and robot presentation method is presented. The implicit function inequalities for representing the outside of the obstacles and the parametric function equalities for representing the surface points of 3D robot are developed. In Section 4, we investigate how to convert the robot path planning problem into a semi-infinite constrained optimization problem. Simulation results are presented in Section 5. Finally conclusions are given in Section 6.

2. Mathematical Background

In this section we will give a brief introduction to various optimization problems, i.e. the standard constrained optimization problem (SCO), generalized constrained optimization problem (GCO), and semi-infinite constrained problem (SICO). The essential part of the mathematical transformation which can transfer a set of inequalities with logic OR operations into one inequality is also introduced in subsection 2.4. Details of the nonlinear

programming theory can be found in (Fletcher, 1987; Gill et al., 1981; Luenberger, 1984; Polak & Mayne, 1984; Rao, 1984; Tanak et al., 1988).

2.1 Optimization Problems

Standard optimization theory (SOT) concerns the minimization or maximization of a function subject to different types of constraints (equality or inequality) (Fletcher, 1987; Gill et al., 1981). There are mainly four different types of optimization problem: *Linear Programming*, *Unconstrained Problems*, *Constrained Problems* and *Semi-infinite Constrained Problems*, as listed in Table 1. The last three parts together comprise the subject of *Non-linear Programming*.

TYPE	NOTATION
Unconstrained scalar	min $f(\boldsymbol{x})$
Unconstrained	min $f(\boldsymbol{x})$
Constrained	min $f(\boldsymbol{x})$ such that $g(\boldsymbol{x}) \le 0$
Goal	min $\boldsymbol{\gamma}$ such that $f(\boldsymbol{x})$-$\boldsymbol{x}^{\boldsymbol{\gamma}} \le \boldsymbol{Goal}$
Minmax	min{max $f(\boldsymbol{x})$} such that $g(\boldsymbol{x}) \le 0$
Nonlinear least squares	min $\sum\{ f(\boldsymbol{x})*f(\boldsymbol{x})\}$
Nonlinear equations	$f(\boldsymbol{x})=0$
Semi-infinite constrained	min $f(\boldsymbol{x})$ such that $g(\boldsymbol{x}) \le 0$ & $\Phi(\boldsymbol{x},\boldsymbol{w}) \le 0$ for all $\boldsymbol{w} \in \Re^2$

Table 1. Types of nonlinear minimization problems.

2.2 Standard Constrained Optimization (SCO)

The standard constrained optimization problem can be described as follows: find an optimal point $\boldsymbol{x}^*$ which minimizes the function:

$$f(\boldsymbol{x}) \tag{1}$$

subject to:

$$g_i(\boldsymbol{x}) = 0,\ i = 1, 2, \ldots, t$$
$$g_j(\boldsymbol{x}) \le 0,\ j = t+1, t+2, \ldots, s$$
$$\boldsymbol{x}_L \le \boldsymbol{x} \le \boldsymbol{x}_U \tag{2}$$

where t and s are positive integers and $s \ge t$, $\boldsymbol{x}$ is an n-dimensional vector of the unknowns $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, and f, g_i $(i = 1, 2, \ldots, t)$ and g_j $(j = t+1, t+2, \ldots, s)$ are real-valued functions of the variables $(x_1, x_2, \ldots, x_n)$. $\boldsymbol{x}_L = (L_1, L_2, \ldots, L_n)$ and $\boldsymbol{x}_U = (U_1, U_2, \ldots, U_n)$ are the lower and upper bounds of $\boldsymbol{x}$, respectively. The function f is the objective function, and the equations and inequalities of (2) are constraints.

It is important to note that although not explicitly stated in the literature available, the logic relationship among the constraints (equalities and inequalities) in (2) are ***logic AND*** (denoted by "$\wedge$"). That is, constraints in (2) can be presented explicitly as:

$$g_1(\boldsymbol{x})=0 \wedge g_2(\boldsymbol{x})=0 \wedge \ldots \wedge g_t(\boldsymbol{x})=0$$
$$\wedge\, g_{t+1}(\boldsymbol{x}) \le 0 \wedge g_{t+2}(\boldsymbol{x}) \le 0 \wedge \ldots \wedge g_s(\boldsymbol{x}) \le 0 \tag{3}$$
$$\wedge\, L_1 \le x_1 \le U_1 \wedge L_2 \le x_2 \le U_2 \wedge \ldots \wedge L_n \le x_n \le U_n.$$

Problem described by (1) and (2) is named as the standard constrained optimization problem (SCOP).

2.3 Generalized Constrained Optimization (GCO)

The work reported in (Wang & Lane, 2000) has shown that some realistic problem can be cast as a generalized constrained optimization problem of the following form:
Find an optimal point $\boldsymbol{x}^*$ which minimizes the function

$$f(\boldsymbol{x}) \tag{4}$$

subject to:

$$\begin{aligned} & g_1(\boldsymbol{x})=0 \wedge g_2(\boldsymbol{x})=0 \wedge \ldots \wedge g_t(\boldsymbol{x})=0 \\ & \wedge g_{t+1}(\boldsymbol{x})\le 0 \wedge g_{t+2}(\boldsymbol{x})\le 0 \ \ldots \wedge g_s(\boldsymbol{x})\le 0 \\ & \wedge (h_{1,1}(\boldsymbol{x})\le 0 \vee h_{1,2}(\boldsymbol{x})\le 0 \vee \ldots \vee h_{1,k_1}(\boldsymbol{x})\le 0) \\ & \wedge (h_{2,1}(\boldsymbol{x})\le 0 \vee h_{2,2}(\boldsymbol{x})\le 0 \vee \ldots \vee h_{2,k_2}(\boldsymbol{x})\le 0) \\ & \wedge \ldots \\ & \wedge (h_{m,1}(\boldsymbol{x})\le 0 \vee h_{m,2}(\boldsymbol{x})\le 0 \vee \ldots \vee h_{m,k_m}(\boldsymbol{x})\le 0) \\ & \wedge L_1\le x_1\le U_1 \wedge L_2\le x_2\le U_2 \wedge \ldots \wedge L_n\le x_n\le U_n \end{aligned} \tag{5}$$

where, the symbol "$\vee$" denotes the ***logic OR*** relationship, $t, s, m, k_1, k_2, ..., k_m$ are all positive integers, and $h_{i,j}(\boldsymbol{x})$, $(i=1,2,\ldots m; j=1,2,\ldots k_i)$, are real-valued functions of the variables $\boldsymbol{x}$. The problem described by (4) and (5) is named as the generalized constrained optimization problem (GCOP) because the constraints have both ***logic AND*** and ***logic OR*** relationships.
The development of an algorithm for the solution to GCOP is important. There are two ways to deal with the difficulty. The first is to develop some new algorithms which can directly deal with the GCOP rather than adopting the algorithms available for the SCOP. The second way is based on the idea of devising a mathematical transformation which is able to convert each constraint: $h_{i,1}(\boldsymbol{x})\le 0 \vee h_{i,2}(\boldsymbol{x})\le 0 \vee \ldots \vee h_{i,k_i}(\boldsymbol{x})\le 0$ $(i=1, 2, \ldots, m)$ into *one* new inequality $H_i(\boldsymbol{x})\le 0$, $i=1, 2, \ldots, m$, for any point $\boldsymbol{x}$. As a result, the algorithms developed for the SCOP can be directly applied to the GCOP.

2.4 A Mathematical Solution to Converting a Set of Inequalities with Logic OR Relation into One Inequality

Here, we present a mathematical transformation which is able to realize the second idea in subsection 2.3. Suppose there are m inequalities $h_i(\boldsymbol{x})<0$, $i=1,2,...,m$, with ***Logic AND*** defined as set A in (6). From a mathematical viewpoint, set A represents the point set of the inside for a generalized n dimensional object, and its complement $\overline{A}$ represents the point set of the outside and boundary of the object. In a 3D space, set definition (6) may be explained as representing the set of all the inside points for an object whose surface is mathematically represented by m continuous equations $h_i(\boldsymbol{x})=0$ $(i=1, 2,..., m)$.

$$A = \{\, \boldsymbol{x} \mid h_1(\boldsymbol{x})<0 \wedge ..... \wedge h_m(\boldsymbol{x})<0 \,\} \tag{6}$$

$$\overline{A} = \{\, \boldsymbol{x} \mid h_1(\boldsymbol{x})\ge 0 \vee h_2(\boldsymbol{x})\ge 0 \vee \ldots \vee h_m(\boldsymbol{x})\ge 0 \,\} \tag{7}$$

For each function $h_i(\boldsymbol{x})$, $(i=1, 2, \ldots, m)$, a new function of the following form is constructed ($\boldsymbol{x}$ is omitted for simplicity):

$$v_i=(h_i^2+t^2)^{1/2}+h_i \qquad i=1, 2, \ldots, m \tag{8}$$

where t is a small, positive real number and satisfies $t<<1$. Note that v_i is the function of a point $\boldsymbol{x}$ and t. For the whole object, a function V of the following form is also constructed:

$$V = v_1 + v_2 + \ldots + v_m = \sum_{i=1}^{m} v_i \tag{9}$$

Now let us examine the properties of the two transformations from h_i to v_i and from v_i to V. First, function v_i is **always positive** for any point $\boldsymbol{x}$ and any constant t, i.e., $v_i>0$ always holds, and second, it is an **increasing function** of h_i, which suggests that the value of v_i at the points where $h_i>0$ is much larger than the value at the points where $h_i<0$. If $t<<1$, v_i can be approximately represented as

$$v_i = \begin{cases} 2h_i + O(t^2) >> t > 0, & h_i > 0; \\ \approx t, & h_i = 0; \\ \approx O(t^2), & h_i < 0. \end{cases} \quad i=1, 2, \ldots, m \tag{10}$$

where $O(t^2)$ represents a very small positive number with the order of t^2 for $t<<1$. (10) indicates that except for the points located at the vicinity of the surface $h_i=0$, v_i is large compared with t when $h_i>0$, and small compared with t when $h_i<0$.

From Fig. 1 we can see that for the points located inside the object and in the vicinity of $h_i=0$, the value of all other functions h_j $(j=1,2,\ldots,m \text{ and } j\neq i)$ is less than zero. This leads to v_i in the order of $O(t^2)$. Substituting (10) into (9) gives

$$V = \begin{cases} >> t, & for \quad (h_1 > 0) \vee (h_2 > 0) \vee \ldots \vee (h_m > 0); \\ \approx t + O(t^2), & for \quad (h_1 = 0 \wedge h_2 \le 0 \wedge h_3 \le 0 \wedge \ldots \wedge h_m \le 0) \vee \\ & \quad (h_2 = 0 \wedge h_1 \le 0 \wedge h_3 \le 0 \wedge \ldots \wedge h_m \le 0) \vee \\ & \quad \ldots \vee (h_m = 0 \wedge h_1 \le 0 \wedge h_2 \le 0 \wedge \ldots \wedge h_{m-1} \le 0); \\ \approx O(t^2), & for \quad (h_1 < 0) \wedge (h_2 < 0) \wedge \ldots \wedge (h_m < 0). \end{cases} \tag{11}$$

Consequently, from (11), (6), and (7) we can observe that: function V is small, $\approx O(t^2)$, compared with t, when all the h_i are sufficiently negative, i.e. at those points which are inside the object; $V>>t+O(t^2)$ at the set of outside points of the object where at least one of the h_i is greater than t; and $V\approx t$ in the vicinity of the boundaries of the object. Fig. 1 illustrates this situation.

Now let us consider the situation when $t\to 0$ to have a better understanding why construction functions (8) and (9) are used as the mathematical transformation. As $t\to 0$, v_i tends to be

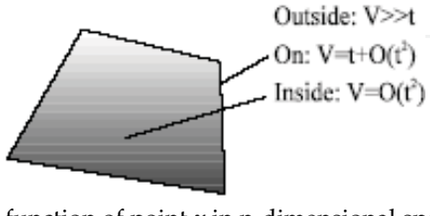


Fig. 1. Illustration of V as a function of point $\boldsymbol{x}$ in n-dimensional space.

$$v_i \begin{cases} > 0, & h_i > 0; \\ = 0, & h_i \le 0. \end{cases} \qquad i=1, 2, \ldots, m \tag{12}$$

It is well-known that the sum of two positive values is positive, the sum of a positive value and a zero is positive, and the sum of two zeroes is zero. Thus the addition operation of v_i in (9) corresponds to the ***Logic OR*** if we treat a positive value as a logic value "1" and a zero as a logic value "0". Thus we have

$$V \begin{cases} > 0, & h_i > 0, \quad for \quad some \quad i \in \{1,2,...,m\}; \\ = 0, & h_i \le 0, \quad for \quad all \quad i \in \{1,2,...,m\}. \end{cases} \tag{13}$$

This property indicates that when t=0 the **sufficient and necessary condition** for a point $\boldsymbol{x}$ which falls into the outside of the object is that

$$V = \sum_{i=1}^{m} v_i > 0 \tag{14}$$

Note that the standard form for constraints in optimization problem (1) and (2) is **less than or equal to** zero. Note that although (14) may change to the form (15), it does not allow the condition "equal to zero".

$$-V = -\sum_{i=1}^{m} v_i < 0 \tag{15}$$

In fact, the "equal to zero" case means the point lies on the boundary of the object. However, it is not desirable for robot path planning to have the path too close to the obstacles. Thus a small positive value Δv can be introduced to control the distance of the feasible path to the obstacle. If the following inequality is satisfied by a point $\boldsymbol{x}$

$$V = \sum_{i=1}^{m} v_i \ge \Delta v \quad \text{or} \quad \Delta v - \sum_{i=1}^{m} v_i \le 0 \tag{16}$$

then this point must be outside the obstacle determined by (6). If $\Delta v \to 0$, the boundary determined by $\Delta v - \sum_{i=1}^{m} v_i \le 0$ tends to be the surface of the obstacle.

In summary, we have the following result.

Theorem 1: *If the outside and the surface of an object is determined by* $(h_1 \ge 0 \vee h_2 \ge 0 \vee \ldots \vee h_m \ge 0)$, *then its outside and surface can also be determined by the inequality* $\Delta v - \sum_{i=1}^{m} v_i \le 0$ *as the small positive value* $\Delta v \to 0$. *In other words, the satisfaction of the inequality* $\Delta v - \sum_{i=1}^{m} v_i \le 0$ *for a point* $\boldsymbol{x}$ *guarantees that this point falls outside the object.*

A direct conclusion drawn from **Theorem 1** is that a GCO problem can be converted into an SCO problem by the transformations (8) and (9).

2.5 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:

Minimize

$$f(\boldsymbol{x}),\ \boldsymbol{x} \in \Re^n, \tag{17}$$

subject to:

$$g_i(\boldsymbol{x}) = 0,\ i = 1, 2, \ldots, t$$
$$g_j(\boldsymbol{x}) \le 0,\ j = t+1, t+2, \ldots, s$$
$$\Phi_k(\boldsymbol{x}, \boldsymbol{v}) \le 0,\ k = 1, 2, \ldots, r$$
$$\boldsymbol{x}_L \le \boldsymbol{x} \le \boldsymbol{x}_U, \text{ for all } \boldsymbol{v} \in \Re^2 \tag{18}$$

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, over which to calculate an appropriately sampled set of values, must be chosen for $\boldsymbol{v}$. $\boldsymbol{x}$ is referred to as the unknown variable and $\boldsymbol{v}$ as the independent variables.

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

(a) Assign an initial point for $\boldsymbol{x}$ and a region for $\boldsymbol{v}$;

(b) Apply a search algorithm to find the optimum solution $\boldsymbol{x}^*$ and the corresponding minimum objective function $f(\boldsymbol{x}^*)$.

In the subsequent sections we will gradually illustrate that the 3D path planning problem without the calculation of Cspace obstacles can be converted into a standard semi-infinite constrained optimization problem.

3. Obstacle and Robot Representations

For robot path planning, the first thing is to give each of the objects a mathematical representation, including obstacles and robot in the workspace. Modeling and manipulation of objects is the research task of Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), and Computer Graphics (CG) (Ricci, 1973; Blechschmidt & Nagasuru, 1990; Barr, 1981; Chiyokura, 1988; Hall & Warren, 1990; Berger, 1986; Comba, 1968; Franklin & Barr, 1981). A solid model should contain an informationally complete description of the geometry and topology of a 3D object (Blechschmidt & Nagasuru, 1990). 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 in 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) (Chiyokura, 1988). In our method, two different categories of obstacles are distinguished, and CSG together with an approximation approach are used to represent the various objects in real world in form of inequality constraints.

3.1 General Representation of Obstacle and Classification

A 3D object $\boldsymbol{S}$ divides the 3D Euclidean space $\boldsymbol{E}^3$ into three parts: the **inside** of the object (denoted by $\boldsymbol{I}$), the **outside** of the object (denoted by $\boldsymbol{T}$), and the **boundary** (denoted by $\boldsymbol{B}$), with

$$\boldsymbol{I} \cup \boldsymbol{B} \cup \boldsymbol{T} = \boldsymbol{E}^3 \tag{19}$$

$$\boldsymbol{I} \cap \boldsymbol{B} = \boldsymbol{B} \cap \boldsymbol{T} = \boldsymbol{I} \cap \mathbf{T} = \Phi \tag{20}$$

Let $\boldsymbol{x}=(x, y, z) \in \boldsymbol{E}^3$ denote a **point** in 3D space. An obstacle can be described as a set of all those points that fall into the inside of the obstacle, that is, an obstacle **A** can be described as:

$$A = \{\, \boldsymbol{x} \mid \boldsymbol{x} \text{ falls into the } \textbf{inside} \text{ of } \textbf{A} \,\} \tag{21}$$

Based on this set-formed representation, we can define an important concept "**free space of an obstacle**" and get a basic condition for a collision-free point.

Definition 1: Free space of an obstacle: The set of points on or outside of the surface of a 3D obstacle is defined as its free space. That is, the **free space** of an obstacle A (set-formed representation) is just $\overline{A}$, i.e., the complement of set A.

Proposition 1: *The necessary and sufficient condition for a point* $\boldsymbol{x}$ *to be collision-free from an obstacle A is that the point* $\boldsymbol{x}$ *must fall into the free space of A, that is,* $\boldsymbol{x} \in \overline{A}$.

The inside of a 3D object can be mathematically represented by one or several united implicit function inequalities. According to the number of the inequalities, we categorize 3D obstacles into two groups.

Definition 2: First group of obstacles: If the inside of a obstacle can be represented by only ***one*** implicit function inequality, the obstacle is said to be in the first group. That is, if an obstacle A can be represented as:

$$A=\{\, \boldsymbol{x} \mid h(\boldsymbol{x}) < 0 \,\} \tag{22}$$

where $h(\boldsymbol{x})$ is an implicit function of $\boldsymbol{x}$, then A belongs to the first group of obstacles. Obviously the free space of A can be represented as the following:

$$\overline{A} = \{\, \boldsymbol{x} \mid h(\boldsymbol{x}) \geq 0 \,\} \tag{23}$$

Examples of the obstacles in the first group include spheres, ellipsoids, torus, superellipsoids and so on (Ricci, 1973; Blechschmidt & Nagasuru, 1990; Barr, 1981; Berger, 1986; Franklin & Barr, 1981; Wang & Lane, 1997). A simple example of the first-group obstacles is illustrated in Fig. 2.

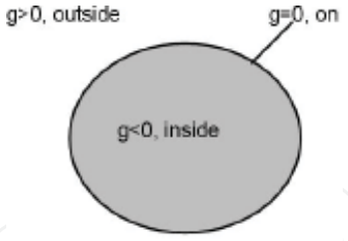


Fig. 2. First group of obstacles: inside and outside of an obstacle.

Definition 3: Second group of obstacles: If the inside of an obstacle must be represented by ***more than one*** united implicit function inequalities, the obstacle is said to be in the second group. That is, if an obstacle A can be represented as:

$$A =\{\, \boldsymbol{x} \mid h_1(\boldsymbol{x})<0 \wedge h_2(\boldsymbol{x})<0 \wedge \ldots \wedge h_m(\boldsymbol{x})<0 \,\} \tag{24}$$

where $h_i(\boldsymbol{x})$, $i=1, 2, \ldots, m$, are all implicit functions of $\boldsymbol{x}$ and m is more than one, then A belongs to the second group of obstacles.

For the second-group obstacle A, the free space can be represented as the following:

$$\overline{A} = \{ \boldsymbol{x} \mid h_1(\boldsymbol{x}) \geq 0 \vee h_2(\boldsymbol{x}) \geq 0 \vee \ldots \vee h_m(\boldsymbol{x}) \geq 0 \} \tag{25}$$

For example, in Fig. 3, the 2-demensioned obstacle is a rectangle whose inside is surrounded by four lines $h_1 = x\text{-}a = 0$, $h_2 = -x\text{-}a = 0$, $h_3 = y\text{-}b = 0$, and $h_4 = -y\text{-}b = 0$, where a and b are positive values. The inside of the obstacle (denoted as A) is the intersection of regions A_i, $i=1, 2, 3, 4$, where each A_i is defined as $A_i = \{ (x, y) \mid h_i < 0 \}$, that is:

$$\begin{aligned} A &= A_1 \cap A_2 \cap A_3 \cap A_4 \\ &= \{ (x, y) \mid h_1 < 0 \} \cap \{ (x, y) \mid h_2 < 0 \} \cap \{ (x, y) \mid h_3 < 0 \} \cap \{ (x, y) \mid h_4 < 0 \} \\ &= \{ (x, y) \mid h_1 < 0 \wedge h_2 < 0 \wedge h_3 < 0 \wedge h_4 < 0 \} \end{aligned} \tag{26}$$

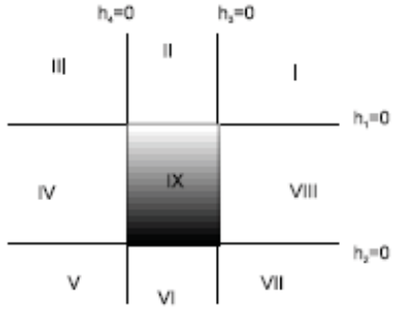


Fig. 3. Second group of obstacles: inside and outside of an obstacle described by **more than one** implicit function inequalities.

where "$\cap$" denotes intersection operation of a set and "$\wedge$" denotes ***logic AND***. The free space of the obstacle (just the union of region I, region II, …, region VIII, and the boundary of the rectangle), can be represented as:

$$\begin{aligned} \overline{A} &= \overline{(A_1 \cap A_2 \cap A_3 \cap A_4)} = \overline{A_1} \cup \overline{A_2} \cup \overline{A_3} \cup \overline{A_4} \\ &= \{ (x, y) \mid h_1 \geq 0 \} \cup \{ (x, y) \mid h_2 \geq 0 \} \cup \{ (x, y) \mid h_3 \geq 0 \} \cup \{ (x, y) \mid h_4 \geq 0 \} \\ &= \{ (x, y) \mid h_1 \geq 0 \vee h_2 \geq 0 \vee h_3 \geq 0 \vee h_4 \geq 0 \} \end{aligned} \tag{27}$$

where "$\cup$" denotes union operation of a set and "$\vee$" denotes ***logic OR***.

3.2 Construction of Object's Defining Inequality

According to section 3.1 we know the inequality "$h(x) < 0$" is a general form to define a representation of an object. We name it as ***defining inequality***. How to construct defining inequality for specific objects in real world? Here we present an approximated method.

To represent an object, another form equivalent to "$h(\boldsymbol{x}) < 0$" is "$f(\boldsymbol{x}) < 1$". The latter form can be easily transformed into "$h(\boldsymbol{x}) < 0$", and is more applicable and convenient for constructing defining inequalities of complex objects from those of the simple objects. In "$f(\boldsymbol{x}) < 1$", the function $f(\boldsymbol{x})$ is named as ***defining function***.

Definition 4: Defining Function: For an object $\boldsymbol{S}$ with its inside $\boldsymbol{I}$, outside $\boldsymbol{T}$, and boundary $\boldsymbol{B}$, a continuous and positive function $f(\boldsymbol{x})$ is called the ***defining function*** of $\boldsymbol{S}$ if for any $\boldsymbol{x} = (x, y, z) \in \boldsymbol{E}^3$, the following hold:

`

$$f(\boldsymbol{x}) = 1 \quad \Leftrightarrow \quad \boldsymbol{x} \in \boldsymbol{B}$$

$$0< f(\boldsymbol{x}) < 1 \;\Leftrightarrow\; \boldsymbol{x} \in \boldsymbol{I} \qquad (28)$$
$$f(\boldsymbol{x}) > 1 \;\Leftrightarrow\; \boldsymbol{x} \in \boldsymbol{T}.$$

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

$$f(\boldsymbol{x}) = (x/R)^2 + (y/R)^2 + (z/R)^2, \qquad (29)$$

and equality $(x/R)^2 + (y/R)^2 + (z/R)^2 = 1$ defines the surface of the sphere.

There are many categories of basic defining functions for object representation (called "primitive solids") such as Quadrics, Superquadrics, and Blobby functions (Berger, 1986).

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

$$x^2 + y^2 + z^2 = r^2 \qquad (30)$$

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

$$(x/r_x)^2 + (y/r_y)^2 + (z/r_z)^2 = 1 \qquad (31)$$

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 numbers.

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

$$\left[r - \sqrt{(x/r_x)^2 + (y/r_y)^2}\;\right]^2 + (z/r_z)^2 = 1 \qquad (32)$$

where r is any given offset value.

b. Superquadrics. This class of objects is a generalization of the quadric representations and provides more flexibility to describe objects (Franklin & Barr, 1981). 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:

$$\left[\left(\frac{x-x_o}{r_x}\right)^{\frac{2}{s_2}} + \left(\frac{y-y_o}{r_y}\right)^{\frac{2}{s_2}}\right]^{\frac{s_2}{s_1}} + \left(\frac{z-z_o}{r_z}\right)^{\frac{2}{s_1}} = 1 \qquad (33)$$

where parameters s_1 and s_2 can be assigned any positive real value. For $s_1=s_2=1$, we get an ordinary ellipsoid. Super-ellipsoid 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

$$\sum_k b_k e^{-a_k r_k^2} = T \tag{34}$$

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.

The defining functions (30) to (34) 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 $x=(x'-a)$ 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 $x'=a$ and with the same thickness of $2a$. More generally, let $M \in R^{3\times3}$ denote the desired rotation matrix and $B=[b_1,b_2,b_3]$ denote the translation vector. Then the translated and rotated solid S is given by:

$$x' = Mx + B \tag{35}$$

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

$$f'(x', y', z') = f(x, y, z) \tag{36}$$

where

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = M^{-1} \begin{pmatrix} x'-b_1 \\ y'-b_2 \\ z'-b_3 \end{pmatrix} \tag{37}$$

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. $M^{-1}=M^T$.

It's easy to give out defining functions for basic object such as sphere and ellipsoids, directly using primitive solids. But objects in real world are usually complex and cannot be directly represented by primitive solids. A natural method to overcome this difficulty is constructing complex objects from simple objects via set operations (union, intersection, and difference).

Given n defining functions $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$, and $f_n(\boldsymbol{x})$ for n objects, respectively, the defining function for the intersection of the n objects is given by

$$f^I(\boldsymbol{x}) = \max(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) \tag{38}$$

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

$$\max(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) = 1 \tag{39}$$

Similarly, the defining function for the union of the n objects is given by

$$f^U(\boldsymbol{x}) = \min(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})). \tag{40}$$

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

$$\min(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) = 1 \tag{41}$$

For example, the intersection of the three infinite slabs with defining functions: $f_1(x)= (x/r)^2$, $f_2(x)= (y/r)^2$, and $f_3(x)= (z/r)^2$ has the following surface equation $\max((x/r)^2, (y/r)^2, (z/r)^2) =1$, which represent the surface of a cube.

Although equations (39) and (41) 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 (39) and (41) 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 (Wang & Cartmell, 1998a), but this method does not apply to non-convex objects. A currently wide-used method is the one reported in (Ricci, 1973). We use that method here. The intersection and union can be smoothly approximated as:

$$f^{I}(x) = (f_1^{m}(x) + f_2^{m}(x) + f_n^{m}(x))^{1/m} \tag{42}$$

$$f^{U}(x) = (f_1^{-m}(x) + f_2^{-m}(x) + f_n^{-m}(x))^{-1/m} \tag{43}$$

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

$$(f_1^{m}(x) + f_2^{m}(x) + f_n^{m}(x))^{1/m} = 1 \tag{44}$$

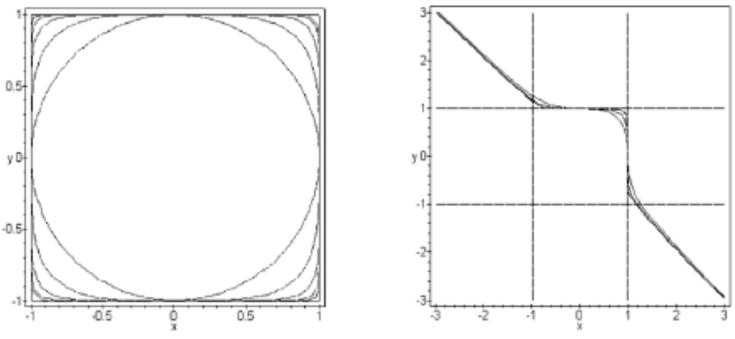
$$(f_1^{-m}(x) + f_2^{-m}(x) + f_n^{-m}(x))^{-1/m} = 1 \tag{45}$$

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 (Ricci, 1973) proved that when $m\rightarrow\infty$, the approximations (42) and (43) give the exact description of the intersection and union respectively. These approximations have the following advantages:

(i) Blending effects are primarily noticeable near surface intersections.

(ii) $f^{I}(x)$ and $f^{U}(x)$ are differentiable which may avoid the possible difficulties in computation due to the differentiability of the max and min functions.

One problem that has been investigated in the previous literature (Wang & Lane, 1997) is the choice of the control parameter m in Equations (44) and (45). Although Ricci (Ricci, 1973) 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.



(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.

Fig. 4. Illustration of intersection $f_1 \cap f_2$. $f_1 = x^2$, $f_2 = y^2$, $((x^2)^m + (y^2)^m)^{1/m} = 1$.

Figures 4(a) and 4(b) show two examples, using two slabs $f_1(x)= x^2$, $f_2(x)= y^2$ as basic primitives to construct a rounded square with different orders (control parameters). In Figure 4(a), m has been chosen to be an integer. When using m=1 to approximate the intersection of $f_1(x)$ and $f_2(x)$, the result is a circle. As m increases the approximation to the intersection of $f_1(x)$ and $f_2(x)$, a square with length and width being 1, get better. It can be seen that when m = 8 or 16, the approximation is very close to the square. In Figure 4(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 $f_1(x)$ and $f_2(x)$. 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 & Barr, 1981). 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 operation. Figures 5 and 6 give two examples. Similarly as m increases, the approximation gets better.

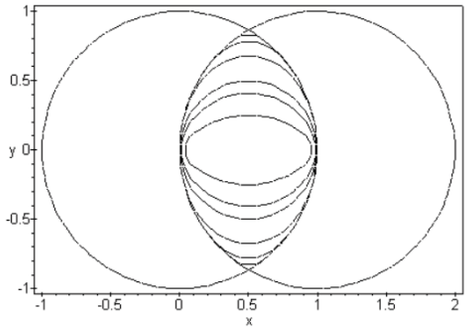


Fig. 5. Illustration of intersection $f_1 \cap f_2$. $f_1 = x^2+y^2$, $f_2 = (x-1)^2+y^2$. m = 0.6, 0.8, 1, 2, 5, and 25 from inside to outside.

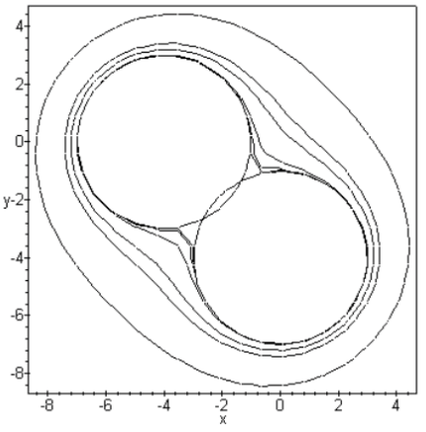


Fig. 6. Illustration of union $f_1 \cup f_2$. $f_1 = [(x+4)^2+y^2]/3^2$, $f_2 = [x^2+(y+4)^2]/3^2$. m= 8, 4, 2, 1, 0.8, and 0.5 from inside to outside.

3.3 Robot Representation

Since robot is a movable and rotatable object in the workspace, to clearly model and dynamically manipulate a robot in 3D space, we must be capable of representing not only its shape and size, but also its spatial location and orientation. Robot representation means the expression of a point, denoted by $\boldsymbol{x}=(x, y, z)$, on the boundary of the robot as the function of its spatial position and orientation variables. Normally there are two mathematical ways to describe the boundary of a robot. The first is the implicit function which takes the form of $g(x, y, z) = 0$ for its boundary expression. The second is the parametric form, in which the x, y, and z are expressed as functions of two auxiliary parameters $\boldsymbol{v}=(t_1, t_2)$, so that $x=\boldsymbol{x}(\boldsymbol{v})=\boldsymbol{x}(t_1, t_2)$, $y=\boldsymbol{y}(\boldsymbol{v})=\boldsymbol{y}(t_1, t_2)$, and $z=\boldsymbol{z}(\boldsymbol{v})=\boldsymbol{z}(t_1, t_2)$. In the following context, we will use both the implicit and the parametric forms to formulate a robot.

Let the geometric centre point $\boldsymbol{O}$ of the robot, denoted by $\boldsymbol{O}(x_o, y_o, z_o)$, be chosen as the position parameters and let a set of three orientation angles, denoted by $\boldsymbol{\Theta}(\theta_1, \theta_2, \theta_3)$, be chosen as the orientation parameters. Then a robot can be represented as:

$$\{ (x, y, z) \mid g(x, y, z, x_o, y_o, z_o, \theta_1, \theta_2, \theta_3) = 0 \} \tag{46}$$

and its equivalent parametric form can be expressed as:

$$\begin{aligned} \{ (x, y, z) \mid & x=\boldsymbol{x}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}), \\ & y=\boldsymbol{y}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}), \\ & z=\boldsymbol{z}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}) \end{aligned} \tag{47}$$

x_o, y_o, z_o together with $\theta_1, \theta_2, \theta_3$ are responsible for determining the position and orientation of a robot. We call $(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3)$ as the robot's ***space configuration variables***.

The implicit function we use to describe the robot here is the superellipsoid proposed in references (Barr, 1981; Berger, 1986), which is a Constructive Solid Geometry (CSG) primitive for a broad family of robot and obstacles. The superellipsoid is defined as:

$$\left[\left(\frac{x-x_o}{r_x} \right)^{\frac{2}{s_2}} + \left(\frac{y-y_o}{r_y} \right)^{\frac{2}{s_2}} \right]^{\frac{s_2}{s_1}} + \left(\frac{z-z_o}{r_z} \right)^{\frac{2}{s_1}} = 1 \tag{48}$$

Its parametric form is:

$$\begin{aligned} & bx= r_x \cos^{s_1}(t_1) \cos^{s_2}(t_2)+ x_o ; \\ & y= r_y \cos^{s_1}(t_1) \sin^{s_2}(t_2)+ y_o; \quad -\pi/2 \le t_1 \le \pi/2; \ 0 \le t_2 \le 2\pi \\ & z= r_z \sin^{s_1}(t_1)+ z_o. \end{aligned} \tag{49}$$

where rx , ry , rz define the geometric extent, s1 and s2 specify the shape properties (s1 is the squareness parameter in the north-south direction; s2 is the squareness parameter in the east-west direction), and xo, yo, zo describe the spatial location. The superellipsoid can be constructed from the basic slabs. Some superellipsoid shapes produced by the choice of different values for parameters s1 and s2 are shown in Fig. 7 when rx=ry=rz.

From (Barr, 1981; Berger, 1986), we know that most kinds of robots can be simulated by the broad family of easily defined superellipsoid primitives. In addition to the superspherical 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 (Barr, 1981; Berger, 1986; Wang & Lane, 1997).

(48) and (49) describe a 3D robot in the standard orientation with $\theta_1=\theta_2=\theta_3=0$. It's 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 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 R is obtained are 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 magnitude a 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 magnitude about the y' axis of frame S' and the resulting frame is called S''. Finally, a positive rotation of magnitude about the z'' axis of S'' is made and the resulting frame is called S'''. Fig. 8 illustrates the combined effect of these steps.
The combined result of these three rotations is mathematically expressed by the following rotation matrix:

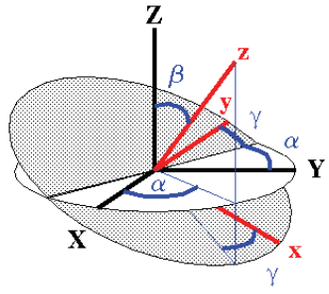


Fig. 8. Euler angle conversion.

$$\begin{pmatrix} x_{new} \\ y_{new} \\ z_{new} \end{pmatrix} = R \cdot \begin{pmatrix} x_{old} \\ y_{old} \\ z_{old} \end{pmatrix} \tag{50}$$

where R is the rotation matrix:

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

(50) is equivalent to (51) as follows:

$$\begin{cases} x_{new} = & x_{old} \cdot (\cos\alpha\cos\beta\cos\gamma - \sin\alpha\sin\gamma) \\ & + y_{old} \cdot (\sin\alpha\cos\beta\cos\gamma + \cos\alpha\sin\gamma) \\ & - z_{old} \cdot (\sin\beta\cos\gamma); \\ y_{new} = & x_{old} \cdot (-\cos\alpha\cos\beta\sin\gamma - \sin\alpha\cos\gamma) \\ & + y_{old} \cdot (\sin\alpha\cos\beta\sin\gamma - \cos\alpha\cos\gamma) \\ & + z_{old} \cdot (\sin\beta\sin\gamma); \\ z_{new} = & x_{old} \cdot (\cos\alpha\sin\beta) + y_{old} \cdot (\sin\alpha\sin\beta) \\ & + z_{old} \cdot \cos\beta; \end{cases} \tag{51}$$

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

$$0 \le a \le 2\pi,\ 0 \le \beta \le \pi,\ 0 \le \gamma \le 2\pi \tag{52}$$

The combination of rotation transformation (52) with (49) results in the parametric form for a superellipsoid robot in a general position $\boldsymbol{O}(x_o, y_o, z_o)$ and orientation $\boldsymbol{\Theta}(\theta_1, \theta_2, \theta_3)$:

$$
\begin{aligned}
x&=(r_x\cos^{s_1}(t_1)\ \cos^{s_2}(t_2))l_1+(r_y\cos^{s_1}(t_1)\ \sin^{s_2}(t_2))l_2+(r_z\sin^{s_1}(t_1))l_3+x_o;\\
y&=(r_x\cos^{s_1}(t_1)\ \cos^{s_2}(t_2))m_1+(r_y\cos^{s_1}(t_1)\ \sin^{s_2}(t_2))m_2+(r_z\sin^{s_1}(t_1))m_3+y_o;\\
z&=(r_x\cos^{s_1}(t_1)\ \cos^{s_2}(t_2))n_1+(r_y\cos^{s_1}(t_1)\ \sin^{s_2}(t_2))n_2+(r_z\sin^{s_1}(t_1))n_3+z_o.
\end{aligned}
\tag{53}
$$

where
where the oriental angles θ_1, θ_2, θ_3 are specified by three Euler angles. For more details about Euler angle conversion, see (Rose, 1957).

4. Converting the Robot Path Planning Problem into the Semi-infinite Optimization Problem

$$
\begin{aligned}
l_1&=cos(\theta_1)cos(\theta_2)cos(\theta_3)-sin(\theta_1)sin(\theta_3)\\
m_1&=-sin(\theta_1)cos(\theta_3)-cos(\theta_1)cos(\theta_2)sin(\theta_3)\\
n_1&=cos(\theta_1)sin(\theta_2)\\
l_2&=-cos(\theta_1)sin(\theta_3)+cos(\theta_2)sin(\theta_2)sin(\theta_3)\\
m_2&=-cos(\theta_1)cos(\theta_3)+sin(\theta_1)cos(\theta_2)sin(\theta_3\\
n_2&=sin(\theta_2)sin(\theta_3)\\
{}_3&=cos(\theta_1)sin(\theta_2)\\
{}_3&=sin(\theta_1)sin(\theta_2)\\
3&=cos(\theta_2)\\
\pi/2&\le t_1\le\pi/2\\
&\le t_2\le 2\pi
\end{aligned}
\tag{54}
$$

4.1 Collision-free Condition for the Obstacle Avoidance

According to **Proposition 1** and the representations of the two classes of obstacles, we can get the condition for a point to be collision-free from an obstacle:

Proposition 2: *The necessary and sufficient condition for a point* $\boldsymbol{x}$ *to be collision-free from a first-group obstacle* $A=\{\boldsymbol{x} \mid h(\boldsymbol{x})<0\}$ *is that* $\boldsymbol{x}\in\overline{A}$, *that is,* $h(\boldsymbol{x})\ge 0$.

Proposition 3: *The necessary and sufficient condition for a point* $\boldsymbol{x}$ *to be collision-free from a second-group obstacle* $A=\{\boldsymbol{x} \mid h_1(\boldsymbol{x})<0 \wedge h_2(\boldsymbol{x})<0 \wedge \ldots \wedge h_m(\boldsymbol{x})<0\}$ *is that* $\boldsymbol{x}\in\overline{A}$, *that is,* $h_1(\boldsymbol{x})\ge 0 \vee h_2(\boldsymbol{x})\ge 0 \vee \ldots \vee h_m(\boldsymbol{x})\ge 0$.

Let $v_i(\boldsymbol{x})=(h_i^2(\boldsymbol{x})+t^2)^{1/2}+h_i(\boldsymbol{x})$, $(i=1, 2, \ldots, m)$, and Δv be a small positive value, then according to **Theorem 1** and **Proposition 3**, we can get a realistically necessary and sufficient condition for a point to be collision-free from a second-group obstacle: $\Delta v-\sum_{i=1}^{m} v_i(x)\le 0$. In fact, with respect to the realistic requirement of robot path planning, we can represent the free space of the second-group obstacle as:

$$
=\{\,\mathbf{x} \mid \Delta v-\sum_{i=1}^{m} v_i(x)\le 0\,\}
\tag{55}
$$

Now let's consider the collision-free condition in the presence of multiple obstacles. In the presence of multiple obstacles, the condition for a point to be collision-free from all obstacles

in the workspace is obviously that the point must fall into the intersection of free spaces of all the obstacles. We get:

Theorem 2: *In the presence of multiple obstacles A_i, i=1, 2,…, j; (j>1), the necessary and sufficient condition for a point $\boldsymbol{x}$ to be collision-free from all the obstacles is that $\boldsymbol{x} \in \overline{A_1} \cap \overline{A_2} \cap ... \cap \overline{A_j}$.*

Suppose in a workspace there are totally s first-group obstacles and m second-group obstacles, and they are respectively defined as:

$$_i = \{ \boldsymbol{x} \mid g_i(\boldsymbol{x}) < 0 \},\ i=1, 2,\ldots, s \tag{56}$$

$$_j = \{ \boldsymbol{x} \mid h_{j,1}(\boldsymbol{x}) < 0 \wedge h_{j,2}(\boldsymbol{x}) < 0 \wedge \ldots \wedge h_{j,k_j}(\boldsymbol{x}) < 0 \},\ j=1, 2, \ldots, m.$$

We note $G_i(\boldsymbol{x}) = - g_i(\boldsymbol{x})$, $i=1, 2,\ldots, s$, and consequently the representation of the free space of A_i changes into:

$$= \{ \boldsymbol{x} \mid G_i(\boldsymbol{x}) \le 0 \},\qquad i=1, 2,\ldots, s. \tag{57}$$

Let $v_{j,r}(\boldsymbol{x}) = (h_{j,r}^2(\boldsymbol{x}) + t^2)^{1/2} + h_{j,r}(\boldsymbol{x})$, ($j=1, 2, \ldots, m$; $r=1, 2,\ldots, k_j$), and Δv_j ($j=1, 2, \ldots, m$) be small positive values. And further let $H_j(\boldsymbol{x}) = \Delta v_j - \sum_{r=1}^{k_j} v_{j,r}(x)$, then free space of B_j can be consequently represented as:

$$\overline{B_j} = \{ \boldsymbol{x} \mid H_j(\boldsymbol{x}) \le 0 \},\qquad j=1, 2, \ldots, m. \tag{58}$$

Thus, the condition for a point $\boldsymbol{x}$ to be collision-free from all the obstacles is that:

$$\in S = \overline{A_1} \cap \overline{A_2} \cap ... \cap \overline{A_s} \cap \overline{B_1} \cap \overline{B_2} \cap ... \cap \overline{B_m}$$

$$\{ \boldsymbol{x} \mid G_1(\boldsymbol{x}) \le 0 \} \cap \{ \boldsymbol{x} \mid G_2(\boldsymbol{x}) \le 0 \} \cap \ldots \cap \{ \boldsymbol{x} \mid G_s(\boldsymbol{x}) \le 0 \}$$

$$\{ \boldsymbol{x} \mid H_1(\boldsymbol{x}) \le 0 \} \cap \{ \boldsymbol{x} \mid H_2(\boldsymbol{x}) \le 0 \} \cap \ldots \cap \{ \boldsymbol{x} \mid H_m(\boldsymbol{x}) \le 0 \}$$

$$\{ \boldsymbol{x} \mid G_1(\boldsymbol{x}) \le 0 \wedge G_2(\boldsymbol{x}) \le 0 \wedge \ldots \wedge G_s(\boldsymbol{x}) \le 0 \wedge H_1(\boldsymbol{x}) \le 0 \wedge H_2(\boldsymbol{x}) \le 0 \wedge \ldots \wedge H_m(\boldsymbol{x}) \le 0 \} \tag{59}$$

that is,

$$G_1(\boldsymbol{x}) \le 0 \wedge G_2(\boldsymbol{x}) \le 0 \wedge \ldots \wedge G_s(\boldsymbol{x}) \le 0 \wedge H_1(\boldsymbol{x}) \le 0 \wedge H_2(\boldsymbol{x}) \le 0 \wedge \ldots \wedge H_m(\boldsymbol{x}) \le 0 \tag{60}$$

4.2 Constraints of Path Planning Problem

An obvious ***necessary and sufficient*** condition for a robot to be collision-free from multiple obstacles is that: all the points inside or on the boundary of the robot fall into the intersection of free spaces of all the obstacles. A little weaker, but sufficient in almost all realistic cases, condition is that: *all the points on the boundary of the robot fall into intersection of all the free spaces.*

Suppose in the workspace there are totally s first-group obstacles and m second-group obstacles, just as defined in (56), and also a static superellipsoid-shaped robot with squareness parameters s_1, s_2, geometric extent r_x, r_y, r_z, center position $\boldsymbol{O}(x_o, y_o, z_o)$ and orientation $\boldsymbol{\Theta}(\theta_1, \theta_2, \theta_3)$. According to (53), the set of the robot boundary points is:

$$T = \{ (x, y, z) \mid$$

$$x = (r_x \cos^{s_1}(t_1) \cos^{s_2}(t_2)) l_1 + (r_y \cos^{s_1}(t_1) \sin^{s_2}(t_2)) l_2 + (r_z \sin^{s_1}(t_1)) l_3 + x_o$$

$$y = (r_x \cos^{s_1}(t_1) \cos^{s_2}(t_2)) m_1 + (r_y \cos^{s_1}(t_1) \sin^{s_2}(t_2)) m_2 + (r_z \sin^{s_1}(t_1)) m_3 + y_o$$

$$z = (r_x \cos^{s_1}(t_1) \cos^{s_2}(t_2)) n_1 + (r_y \cos^{s_1}(t_1) \sin^{s_2}(t_2)) n_2 + (r_z \sin^{s_1}(t_1)) n_3 + z_o \quad -\pi/2 \le t_1 \le \pi/2;\ 0 \le t_2 \le 2\pi. \tag{61}$$

l_i, m_i, n_i, (i=1,2,3) are defined same as in (54). Let (x, y, z) = ($\boldsymbol{X}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, t_1, t_2)$, $\boldsymbol{Y}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, t_1, t_2)$, $\boldsymbol{Z}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, t_1, t_2)$), and the space configuration vector $\boldsymbol{u}$= (x_o, y_o, z_o, θ_1, θ_2, θ_3), then the collision-free condition, according to (60), can be represented as:
For all $\boldsymbol{v}$ = (t_1, t_2) that $-\pi/2 \le t_1 \le \pi/2$, $0 \le t_2 \le 2\pi$, the following expression hold:

$$G_1(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), Z(\boldsymbol{u}, \boldsymbol{v})) \le 0 \wedge$$
$$G_2(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \wedge \ldots \wedge$$
$$G_s(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \wedge$$
$$H_1(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \wedge$$
$$H_2(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \wedge \ldots \wedge$$
$$H_m(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \tag{62}$$

Let P_i ($\boldsymbol{u}$, $\boldsymbol{v}$) = $G_i(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v}))$ i=1,2,…,s; Q_j ($\boldsymbol{u}$, $\boldsymbol{v}$) =H_j ($\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})$) j = 1,2,…, m, then (62) changes into the following:

$$P_1(\boldsymbol{u}, \boldsymbol{v}) \le 0 \wedge P_2(\boldsymbol{u}, \boldsymbol{v}) \le 0 \wedge \ldots \wedge P_s(\boldsymbol{u}, \boldsymbol{v}) \le 0 \wedge$$
$$Q_1(\boldsymbol{u}, \boldsymbol{v}) \le 0 \wedge Q_2(\boldsymbol{u}, \boldsymbol{v}) \le 0 \wedge \ldots \wedge Q_m(\boldsymbol{u}, \boldsymbol{v}) \le 0 \tag{63}$$

In path planning the configuration variables also have certain range limits. This requirement can be described as:

$$x_L \le x_o \le x_U,\ y_L \le y_o \le y_U,\ z_L \le z_o \le z_U,\ \alpha_1 \le \theta_1 \le \beta_1,\ \alpha_2 \le \theta_2 \le \beta_2,\ \alpha_3 \le \theta_3 \le \beta_3 \tag{64}$$

that is,

$$\boldsymbol{u_L} \le \boldsymbol{u} \le \boldsymbol{u_U} \tag{65}$$

where $\boldsymbol{u_L}$=(x_L, y_L, z_L, α_1, α_2, α_3), $\boldsymbol{u_U}$=(x_U, y_U, z_U, β_1, β_2, β_3).
(63) and (65) form the inequality constraints required in the formulation of the SICO problem for path planning.

4.3 Design of the Objective Function

There are many ways to design the objective function. In a 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 the form (66) as the objective function, with the goal configuration point (x_g, y_g, z_g) and goal configuration angles (φ_1, φ_2, φ_3) being its unique global minimum point and satisfying the condition that $min\ f(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3) = f(x_g, y_g, z_g, \varphi_1, \varphi_2, \varphi_3) = 0$.

$$f(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3) =$$
$$w((x_o-x_g)^2+(y_o-y_g)^2+(z_o-z_g)^2)+(1-w)((\theta_1-\varphi_1)^2+(\theta_2-\varphi_2)^2+(\theta_3-\varphi_3)^2) \tag{66}$$

where w is a non-negative weighted factor that satisfies: $0 \le w \le 1$.
In (66), w is used to adjust the relative effects of the spatial position $(x_o-x_g)^2+(y_o-y_g)^2+(z_o-z_g)^2$ and the spatial orientation $(\theta_1-\varphi_1)^2+(\theta_2-\varphi_2)^2+(\theta_3-\varphi_3)^2$. When w=1, only the effect of the spatial position factor is considered. w can be adaptively adjusted and be revised during the searching process. (66), (63), and (65) together form a semi-infinite constrained optimization

problem. If we use the initial configuration variables $(x_s, y_s, z_s, \delta_1, \delta_2, \delta_3)$ 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 $x_o^* = x_g, y_o^* = y_g, z_o^* = z_g, \theta_1^* = \varphi_1, \theta_2^* = \varphi_2, \theta_3^* = \varphi_3$.
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 3D 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 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. The vehicle is modeled as a superellipsoid with different shapes and the obstacles are modeled by circle, ellipsoid, cylinder, tetrahedron, cuboids, and various other shapes of superellipsoid, which belong to either the first group or the second group.

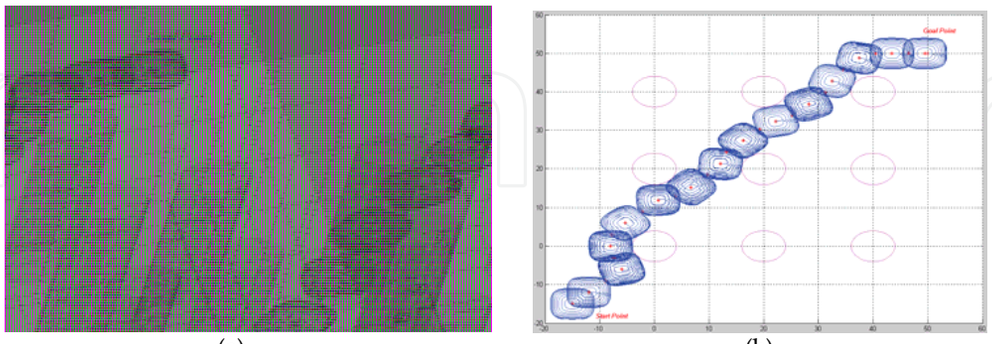
5.1 Algorithm Implementation Consideration

The implementation of the semi-infinite optimization is based on the constrained optimization toolbox (The Math Works Inc., 1993), but some modifications have been made. The first is the control of the output. In (The Math Works Inc., 1993), 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 the 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 nonlinear programming is to minimize the number of function evaluations, which represents the most efficient way for finding the optimum $\boldsymbol{x}^*$. Thus, the changing 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 (The Math Works Inc., 1993).
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) (The Math Works Inc., 1993). The limits of the workspace are: Ol=(-20, -20, -20) and Ou=(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 Ol≤O≤Ou will guarantee that the generated path must be in the workspace. Let Os=(xs, ys, zs) and Θs=(δ1, δ2, δ3) denote the initial configuration and Og=(xg, yg, zg), Θg=(φ1, φ2, φ3) denote the goal configuration. The robot's start and goal configurations are chosen as (xs, ys, zs, δ1, δ2, δ3)=(-20, -20, -20, 0, 0, 0) and (xg, yg , zg , φ1, φ2, φ3)=(50, 50, 50, 0, 0, 0) respectively . The objective function is defined as:

$$f = w((x_o-50)^2+(y_o-50)^2+(z_o-50)^2)+(1-w)(\theta_1^2+\theta_2^2+\theta_3^2) \quad (67)$$

where w is chosen as $0 \le w \le 1$.

5.2 Simulation Results with the First Kind of Objects



(a) (b)

Fig. 10. Workspace with nine cylinders and the generated path. (a) 3D view. (b) 2D view.

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 Figs. 9(a) and 9(b). The obstacles are represented as spheres with the following boundary expressions: $(x-20)^2 + (y-20)^2 + (z-20)^2 = 225$, $x^2 + (y-20)^2 + (z-20)^2 = 25$, $(x-40)^2 + (y-20)^2 + (z-20)^2 = 25$, $(x-20)^2 + y^2 + (z-20)^2 = 25$, $(x-20)^2 + (y-40)^2 + (z-20)^2 = 25$, $(x-20)^2 + (y-20)^2 + z^2 = 25$, $(x-20)^2 + (y-20)^2 + (z-40)^2 = 25$.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. Figs. 9(a) and 9(b) show the same generated path, the same obstacles and the same robot, but from different view angles. It can be observed that the optimization algorithm does converge to the goal and a smooth path has been generated.

In order to illustrate the suitability of the approach for different obstacle shapes, we have shown another distributed-obstacle situation as shown in Figs. 10(a) and 10(b). The results simulate a real world path planning task 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: $x^2 + y^2 = 16$, $x^2 + (y-20)^2 = 16$, $x^2 + (y-40)^2 = 16$, $(x-20)^2 + y^2 = 16$, $(x-20)^2 + (y-20)^2 = 16$, $(x-20)^2 + (y-40)^2 = 16$, $(x-40)^2 + y^2 = 16$, $(x-40)^2 + (y-20)^2 = 16$, $(x-40)^2 + (y-40)^2 = 16$ where $-20 \le z \le 60$. 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 superellipsoid. Fig. 10(a) is a 3D view and Fig. 10(b) is a 2D view. In this example the 2D view clearly shows the collision avoidance of the robot from the obstacles. From Figs. 10(a) and 10(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.

In addition to the simulation results shown in Figs. 9 and 10, we have carried out another test with mixed superellipsoids and cylinders as shown in Figs. 11(a) and 11(b). 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: 2 cylinders : $(x-40)^2 + y^2 = 144$, $30 \le z \le 50$, $(x-30)^2 + (y-30)^2 = 100$, $-5 \le z \le 30$, 3 superellipoids: $((x-15)/12) + ((y+10)/8) + ((z-12)/12) = 1$, $((x-15)/20)^{2/3} + ((y-40)/18)^{2/3} + ((z+10)/12)^{2/3} = 1$, $(x-10)^2 + (y-10)^2 + (z-10)^2 = 100$. Fig. 11(b) is an enlarged view of Fig. 11(a). The experimental results show that the robot can adjust its orientation angles autonomously to reach the goal point.

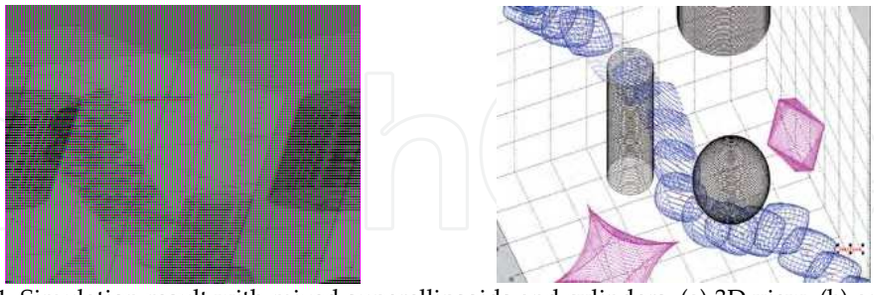
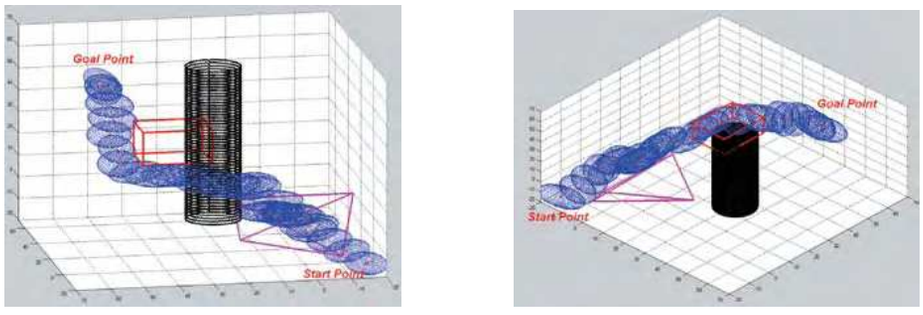


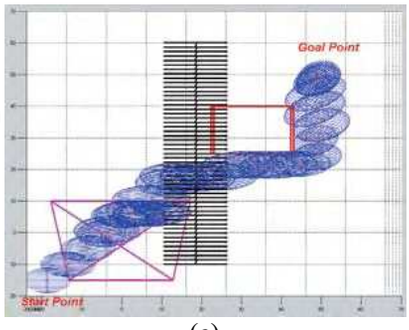
Fig. 11. Simulation result with mixed superellipsoids and cylinders. (a) 3D view. (b) another 3D view.

5.3 Simulation Results with Both the First and Second Kinds of Objects

Figs. 12(a), 12(b), and 12(c) show the experimental results with the environment including a triangular pyramid and a cube which belong to the second group and a cylinder-like obstacle belonging to the first group. The triangular pyramid's four vertexes are: V_0=(-5,-10,-15), V_1=(20,15,-15), V_2=(5,-15,10), V_3=(5,20,10) and its outside is represented by $\{(h_1 \geq -x-y-0.6z-14) \cup (h_2 \geq -x+0.4z+1) \cup (h_3 \geq x+0.6z-11) \cup (h_4 \geq -x+y-0.8z-7)\}$.The cube is represented by $10 \leq x \leq 25$, $25 \leq y \leq 45$, $25 \leq z \leq 40$, while the cylinder is descried as $(x-35)^2 + (y-20) \leq 64$, $-10 \leq z \leq 50$. The robot is represented by a superellipsoid with r_x=8, r_y=6, r_z=4, s_1=s_2=1.It is obvious that the path generated clearly avoids the obstacle and converges to the goal.



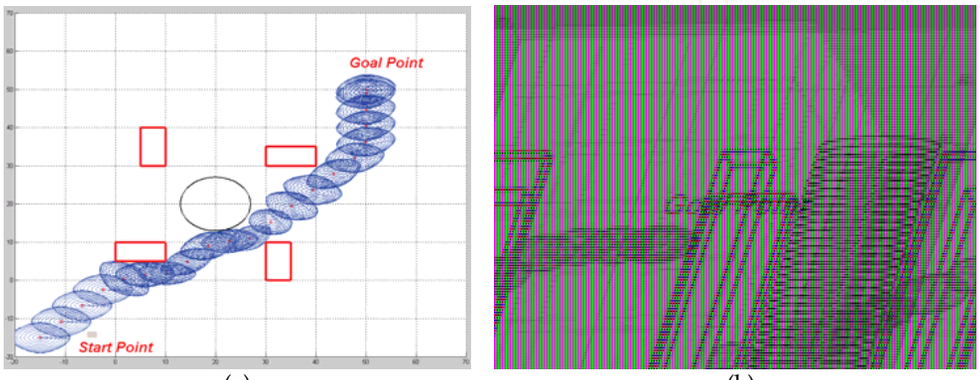
(a) (b)



(c)

Fig. 12. Simulation result with both the first and the second kinds of objects. (a) 3D view. (b) another 3D view. (c) 2D view.

Figs. 13(a) and 13(b) show the results simulating another typical real world path planning task encountered in offshore industry. The designed workspace contains a cylinder and several cubes which may be models for pipelines or a base of an offshore structure. The cylinder is described as follows: $(x-20)^2 + (y-20)^2 \leq 49, -10 \leq z \leq 50$. The four cubes are represented as: $\{0 \leq x \leq 10, 5 \leq y \leq 10, -15 \leq z \leq 55\}$, $\{30 \leq x \leq 40, 30 \leq y \leq 35, -15 \leq z \leq 55\}$,$\{5 \leq x \leq 10, 30 \leq y \leq 40, -15 \leq z \leq 55\}$,$\{30 \leq x \leq 35, 0 \leq y \leq 10, -15 \leq z \leq 55\}$. The robot is represented by a superellipsoid with $r_x=6$, $r_y=4$, $r_z=3$, $s_1=s_2=1.2$. Figs. 13(a) and 13(b) show the clear avoidance of the obstacles by the robot and the convergence to the goal.



(a) (b)

Fig. 13. Simulation result with four cubes and one cylinder. (a) 2D view. (b) 3D view.

5.4 Discussion about Efficiency of the Approach

A criterion for efficiency analysis used by many previous path planning algorithms is the worst case computational complexity when the robot path planning problem is formulated as a discrete mathematics problem (Kreyszig, 1993; Rosen, 1991), based on the assumption that objects are represented by polygons or polyhedrons and a discrete search algorithm is used. However in constrained optimization there is no such a criterion corresponding to that used in discrete mathematics for measuring the worst case computational complexity, because the time used depends on both the size of the problem (i.e. the number of the equalities) and the form of the objective function and the constraints.

As shown in our simulation experiments, the time for the robot to take a step varies, because at some places it must adjust its pose or step size, and at other places it doesn't have to. In the simulation experiment shown in Figs. 10 (a) and 10(b), the average time for one step is about 0.8s with Intel Pentium 4 CPU running the algorithm realized in Matlab. Besides improving the performance of the implementation and the algorithm itself, there is still big room for efficiency improvement. For example, in our experiments the value of the distance that robot is permitted to be close to obstacles is so small (close to 0) that sample points of the surface of the robot (represented as a superellipsoid) and the obstacles must be very dense, otherwise collisions are likely to occur. Density of sample points is one of the main sources of the computational complexity, but in real applications, the distance is usually bigger and the sample points may be sparse, and consequently the time needed for computation may be much less.

To precisely analyze the computational complexity of our method is an important, necessary, but difficult job. It must include determining proper complexity description model,

analyzing performance of the implementation of SCO algorithm which is applied in our method, etc. We hope to address it in further papers.

6. Conclusions

In this chapter, inequality transformation and semi-infinite constrained optimization techniques have been presented for the development of a realistic Robot Path Planning approach. 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. From the viewpoint of robot path planning, this paper presents a new way of using a classical engineering approach. The generality of representing the free space of all the objects using the inequalities $g_i(x) \leq 0$ makes this optimization-based approach suitable for different object shapes, and it has significantly simplified the construction of the objects by sensors and computer vision systems. The iterative nature of the search for the optimization point makes it particularly suitable for on-line sensor-based path planning. Once a new object is detected, a new inequality can be added before running the next iteration. When an old obstacle is passed, its corresponding inequality may also be deleted. Every time when an inequality is added or deleted, a new optimization problem is formed. The current variable is always treated as the initial point of the optimization problem and search starts again. This mechanism indicates this approach is efficient and particularly suitable for on-line path planning. Other advantages of the approaches include that mature techniques developed in nonlinear programming theory with guarantee of 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:

1) The robot does not need to be shrunk to a point, the obstacles do not need to be expended to Cspace, and the entire course of path planning can be carried out in real 3D space.
2) The goal point is guaranteed to be the only global minimum of the objective function.
3) The standard search techniques which have been developed for more than thirty years in the nonlinear programming field can be used.
4) The approach is suitable for on-line task planning.

The investigation carried out in this chapter has also indicated that robot path planning can be formulated in different ways. It's important to seek more efficient and realistic methods for problem formalization. Computational complexity analysis must be developed based on a proper problem formulation which considers enough constraints. Although the fundamentals for the nonlinear programming theory have existed for many years, they have not attracted enough attention for such applications. The context presented in this chapter 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 chapter has shown its great potential as an on-line motion planner. The future work includes the extension of the principle developed here to the obstacle avoidance problem for manipulator without the calculation of Cspace obstacles, and the adoption of the interpolation techniques to deal with the local minima problem (Wang et al., 2000).

The constraints added by the kinematics and the shape of a manipulator are more complex than the subsea vehicle we have done in this chapter. In addition, for a car_like moving

robot, the nonholonomic constraints must be taken into account. Path planning without the Cspace computation and with the consideration of those practical issues is still the challenge we are facing.

7. 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, the National High-Tech Development 863 Program of China under Grant No. 2003AA1Z2220, the National Natural Science Fouondation of China under Grant No. 60373053, and the financial support from Chinese Academy of Sciences and Chinese State Development Planning Commission for Bai Ren Ji Hua (BRJH), 2002-2005.

8. References

Barr, A. (1981). Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1, 1, (January 1981), 11-23, ISSN: 0272-1716.

Berger, M. (1986). *Computer Graphics with Pascal*. Benjamin-Cummings Publishing Co., Inc., ISBN: 0805307915, California.

Blackmore, D.; Leu, M. & Wang, L. (1997). The sweep-envelope differential equation algorithm and its application to NC machining verification. *Computer Aided Design*, 29, 9, (September 1997), 629-637, ISSN: 0010-4485.

Blechschmidt, J. & Nagasuru, D. (1990). The use of algebraic functions as a solid modelling alternative: an investigation. *Proceedings of the 16th ASME Design Automation Conference*, pp. 33-41. Chicago, September 1990, ASME, New York.

Brooks, R. & Lozano-Perez, T. (1985). A subdivision algorithm in configuration space for findpath with rotation. *IEEE Trans. on Systems, Man and Cybernetics*, 1985, 15, 2, (March 1985), 224-233, ISSN: 1083-4427.

Chang, J. & Lu, H. (2001). Backing up a simulated truck via grey relational analysis. *Journal of the Chinese Institute of Engineers*, 24, 6, (November 2001), 745-752, ISSN: 0253-3839.

Chiyokura, H. (1988). *Solid Modelling with Designbase: Theory and Implementation*. Addison-Wesley Publishing Limited, ISBN: 0-201-19245-4, New York.

Comba, P. (1968). A procedure for detecting intersections of three-dimensional objects. *JACM*, 15, 3, (July 1968), 354-366, ISSN: 0004-5411.

Conn, R. & Kam, M. (1997). On the moving-obstacle path planning algorithm of Shih, Lee, and Gruver. *IEEE Trans. Systems, Man and Cybernetics, Part B: Cybernetics*, 27, 1, (February 1997), 136-138, ISSN: 1083-4419.

Connolly, I. (1997). Harmonic function and collision probabilities. *Int. J. Robotics Research*, 16, 4, (August 1997), 497-507, ISSN: 0278-3649.

Fletcher, R. (1987). *Practical Methods of Optimization*. John Wiley & Sons, Inc., ISBN: 0-471-91547-5, New York.

Franklin, W. & Barr, A. (1981). Faster calculation of superquadric shapes. *IEEE Computer Graphics & Application*, 1, 3, (July 1981), 41-47, ISSN: 0272-1716.

Gill, P.; Murray, W. & Wright, M. (1981). *Practical Optimization*. Academic Press, ISBN: 0-12-283952-8, London.

Hall, M. & Warren, J. (1990). Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics & Applications*, 10, 6, (November 1990): 33-42, ISSN: 0272-1716.

Haug, E.; Adkins, F. & Cororian, D. (1998). Domains of mobility for planar body moving among obstacles. *Trans. the ASME, Journal of Mechanical Design*, 120, 3, (September 1998), 462-467, ISSN: 1050-0472.

Hu, T.; Kahng, A. & Robins, G. (1993). Optimal robust path planning in general environment. *IEEE Trans. Robotics and Automation*, 9, 6, (December 1993), 755-774, ISSN: 1042-296X.

Huang, H. & Lee, P. (1992). A real-time algorithm for obstacle avoidance of autonomous mobile robots. *Robotica*, 10, 3, (May 1992), 217-227, ISSN: 0263-5747.

Hwang, Y. & Ahuja, N. (1992). A potential field approach to path planning. *IEEE Trans. on Robotics and Automation*, 8, 1, (February 1992), 23-32, ISSN: 1042-296X.

Khatib, O. (1986). Real-time obstacle avoidance for manipulator and mobile robots. *Int. J. Robotics Research*, 5, 1, (February 1986): 90-98, ISSN: 0278-3649.

Kreyszig, E. (1993). *Advanced Engineering Mathematics*. John Wiley & Sons, Inc., ISBN: 0471728977, New York.

Latombe, J. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, ISBN: 0-7923-9206-X, Boston.

Lozano-Perez, T. (1983). Spatial planning: A configuration space approach. *IEEE Trans. on Computers*, 32, 2, (February 1983), 108-120, ISSN: 0018-9340.

Lu, H. & Yeh, M. (2002). Robot path planning based on modified grey relational analysis. *Cybernetics and Systems*, 33, 2, (March 2002), 129-159, ISSN: 0196-9722.

Luenberger, D. (1984). *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, ISBN: 0-201-15794-2, London.

Lumelsky, V. (1991). A comparative study on path length performance of maze-searching and robot motion planning. *IEEE Trans. Robotics and Automation*, 7, 1, (February 1991), 57-66, ISSN: 1042-296X.

Oriolo, G.; Ulivi, G. & Vendittelli, M. (1998). Real-time map building and Navigation for autonomous robots in unknown environments. *IEEE Trans. Systems, Man and Cybernetics, PartB: Cybernetics*, 28, 3, (June 1998), 316-333, ISSN: 1083-4419.

Park, T.; Ahn, J. & Han, C. (2002). A path generation algorithm of an automatic guided vehicle using sensor scanning method. *KSME International Journal*, 16, 2, (February 2002), 137-146, ISSN:1226-4865.

Petillot, Y.; Ruiz, I. & Lane, D. (2001). Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar. *IEEE Journal of Oceanic Engineering*, 26, 2, (April 2001), 240-251, ISSN: 0364-9059.

Petillot, Y.; Ruiz, T.; Lane, D.; Wang, Y.; Trucco, E. & Pican, N. (1998). Underwater vehicle path planning using a multi-beam forward looking sonar. *IEEE Oceanic Engineering Society, OCEANS'98*, pp. 1194-1199, ISBN: 0-7803-5045-6, Nice, France, September 1998, IEEE Inc., Piscataway.

Polak, E. & Mayne, D. (1984). Control system design via semi-infinite optimization: a review. *Proceeding of the IEEE*, 72, 12, (December 1984): 1777-1793, ISSN: 0018-9219.

Rao, S. (1984). *Optimization, theory and applications*. John Wiley & Sons, Inc., ISBN: 0-470-27483-2, New York.

Ricci, A. (1973). A constructive geometry for computer graphics. *The Computer Journal*, 16, 2, (April 1973), 157-160, ISSN: 0010-4620.

Rose, E. (1957). *Elementary Theory of Angular Momentum*. John Wiley & Sons, Inc., ISBN: 0471735248, New York.

Rosen, K. (1991). *Discrete mathematics and Its Applications*. McGraw-Hill, Inc., ISBN: 0-07-053744-5, New York.

Ruiz, T.; Lane, D. & Chantler, M. (1999). A comparison of inter-frame feature measures for robust object classification in sector scan sonar image sequences. *IEEE Journal of Oceanic Engineering*, 24, 4, (October 1999), 458-469, ISSN: 0364-9059.

Shih, C. & Jeng, J. (1999). Stabilization of non-holonomic chained systems by gain scheduling. *International Journal of Systems Science*, 30, 4, (April 1999), 441-449, ISSN: 0020-7721.

Sundar, S. & Shiller, S. (1997). Optimal obstacle avoidance based on the Hamilton-Jacobi-Bellman equation. *IEEE Trans. Robotics and Automation*, 13, 2, (April 1997), 305 -310, ISSN: 1042-296X.

Tanak, Y.; Fukushima, M. & Ibaraki, T. (1988). A comparative study of several semi-infinite nonlinear programming algorithms. *European Journal of Operational Research*, 36, 1, (July 1988), 92-100, ISSN: 0377-2217.

The Math Works Inc. (1993). *MATLAB Optimization Toolbox User's Guide*.

Trucco, E.; Petillot, Y.; Ruiz, I.; Plakas, K. & Lane, D. (2000). Feature tracking in video and sonar subsea sequences with applications. *Computer Vision and Image Understanding*, 79, 1, (July 2000), 92-122, ISSN: 1077-3142.

Wang, W. & Wang, K. (1986). Geometric Modeling for swept volume of moving solids. *IEEE Computer Graphics & Applications*, 6, 12, (December 1986), 8-17, ISSN: 0272-1716.

Wang, Y. (1995). *Kinematics, motion analysis and path planning for four kinds of wheeled mobile robots* [Dissertation]. Dept. Mechanical Engineering, Edinburgh University.

Wang, Y. (1997). A note on solving the find-path problem by good representation of free space. *IEEE Trans. Systems, Man and Cybernetics, Part B: Cybernetics*, 27, 1, (February 1997), 723-724, ISSN: 1083-4419.

Wang, Y. & Cartmell, M. (1998a). Autonomous vehicle parallel parking design using function fitting approaches. *Robotica*, 16, 2, (March 1998), 159-170, ISSN: 0263-5747.

Wang, Y. & Cartmell, M. (1998b). Trajectory generation for four-wheel steering tractor-trailer system: a two step method. *Robotica*, 16, 4, (July 1998), 381-386, ISSN: 0263-5747.

Wang, Y. & Cartmell, M. (1998c). New Model for Passing Sight Distance on Two-Lane Highways. *ASCE J. of Transportation Engineering*, 124, 6, (November 1998), 536-545, ISSN: 0733-947X.

Wang, Y.; Cartmell, M.; Tao, Q. & Liu, H. (2005). A generalized real-time obstacle avoidance method without the Cspace calculation. *J. Computer Science & Technology*, 20, 6, (November 2005), 774-787, ISSN: 1000-9000.

Wang, Y. & Lane, D. (1997). Subsea vehicle path planning using nonlinear programming and constructive solid geometry. *IEE Proc., Part D, Control theory and applications*, 144, 2, (March 1997), 143-152, ISSN: 1350-2379.

Wang, Y. & Lane, D. (2000). 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. Systems, Man and Cybernetics, Part C: Application and Reviews*, 30, 4, (November 2000), 525-536, ISSN: 1094-6977.

Wang, Y.; Lane, D. & Falconer, G. (2000). Two novel approaches for unmanned underwater vehicle path planning: constrained optimization and semi-infinite constrained optimization. *Robotica*, 18, 2, (March 2000), 123-142, ISSN: 0263-5747.

Wang, Y. & Linnett, J. (1995). Vehicle kinematics and its application to highway design. *ASCE J. of Transportation Engineering*, 121, 1, (January 1995), 63-74, ISSN: 0733-947X.

Wang, Y.; Linnett, J. & Roberts, J. (1994a). Motion feasibility of a wheeled vehicle with a steering angle limit. *Robotica*, 12, 3, (May 1994), 217-226, ISSN: 0263-5747.

Wang, Y.; Linnett, J. & Roberts, J. (1994b). Kinematics, kinematic constraints and path planning for wheeled mobile robots. *Robotica*, 12, 5, (September 1994), 391-400, ISSN: 0263-5747.

Wang, Y.; Liu, H.; Li, M.; Wang, Q.; Zhou, J. & Cartmell, M. (2004). A real-time path planning approach without the computation of Cspace obstacles. *Robotica*, 22, 2 (March 2004), 173-187, ISSN: 0263-5747.

Xu, W. & Ma, B. (1999). Polynomial motion of non-holonomic mechanical systems of chained form. *Mathematical Methods in the Applied Sciences*, 22, 13, (September 1999), 1153-1173, ISSN: 0170-4214.

Zhang, Y. & Valavanis, K. (1997). A 3-D potential panel method for robot motion planning. *Robotica*, 15, 4, (July 1997), 421-434, ISSN: 0263-5747.

Mobile Robots: Perception & Navigation

Edited by Sascha Kolski

ISBN 3-86611-283-1

Hard cover, 704 pages

Publisher Pro Literatur Verlag, Germany / ARS, Austria

Published online 01, February, 2007

Published in print edition February, 2007

Today robots navigate autonomously in office environments as well as outdoors. They show their ability to beside mechanical and electronic barriers in building mobile platforms, perceiving the environment and deciding on how to act in a given situation are crucial problems. In this book we focused on these two areas of mobile robotics, Perception and Navigation. This book gives a wide overview over different navigation techniques describing both navigation techniques dealing with local and control aspects of navigation as well es those handling global navigation aspects of a single robot and even for a group of robots.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Yongji Wang, Matthew Cartmell, QingWang and Qiuming Tao (2007). A Generalized Robot Path Planning Approach Without The Cspace Calculation, Mobile Robots: Perception & Navigation, Sascha Kolski (Ed.), ISBN: 3-86611-283-1, InTech, Available from: http://www.intechopen.com/books/mobile_robots_perception_navigation/a_generalized_robot_path_planning_approach_without_the_cspace_calculation

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen