# Efficient Algorithm for Collision Detection and Path Planning for Robotics Application

by

Mohammad Dikko S. Aliyu

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

**MASTER OF SCIENCE**

In

**SYSTEMS ENGINEERING**

June, 1994

Order Number 1360404

# Efficient algorithm for collision detection and path planning for robotics application

Aliyu, Mohammad Dikko S., M.S.

King Fahd University of Petroleum and Minerals (Saudi Arabia), 1994

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

# Efficient Algorithm for Collision Detection and Path Planning for Robotics Application

BY

Mohammad Dikko S. Aliyu

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

**KING FAHD UNIVERSITY OF PETROLEUM & MINERALS**

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

## MASTER OF SCIENCE
## In
## Systems Engineering

June, 1994

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN 31261, SAUDI ARABIA

COLLEGE OF GRADUATE STUDIES

This thesis , written by Mohammad Dikko S. Aliyu under the direction of his Thesis advisor and approved by his Thesis Commitee, has been presented to and accepted by the Dean of the College of Graduate Studies, in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

Thesis Commitee:

K. Al-Sultan, 5/6/96
*Dr. Khaled Saleh Al – Sultan (Advisor)*

M. Reyhanoglu
*Dr. Mahmut Reyhanoglu (Member)*

*Dr. Mohammad Ben Daya(Member)*

*Dr. Shokri Z. Selim (Member)*

K. Al-Sultan, 5/6/96
*Chairman Systems Engineering*

*Dean, College of Graduate Studies*

5 6 96
*Date*

كلية الدراسات العليا
COLLEGE OF GRADUATE STUDIES

# ACKNOWLEDGEMENTS

After praise and thanks to Allah (SWA) for his help and beneficence in this thesis, and salutations to his noble prophet Muhammad (peace and blessings of Allah upon him), I wish to express my profound gratitude to my advisor, Dr. Khalid S. Al-Sultan, for the initiation, guidance, support and encouragement that he provided throughout all the stages of this thesis.

I am greatly indebted to Dr. Shokri Z. Selim, Dr. Mohammad Ben Daya, and Dr. Mehmut Reyhanoglu for the significant contribution, untiring guidance and encouragement that they have provided in this thesis.

The encouragement and good wishes of the following friends in Systems Engineering Dept. and other Depts. is also worthy of acknowledgement. They are: Mr. Wasim Al-Baroudi, Mr. Samir Al-Amer, Mr. Imran Tasadduq, Mr. Asim Humayun, Mr. Farooq Anjum, Mr. Saleh Al-Rumaih, Mr. Ma'aroof Khan, Mr. Jibril Odogba, Mr. Moh'd Al-Fawzan, Mr. Anwarul Islam, Mr. Ammar Abdu, Mr. Farruk Pulak, Mr. Anas Vaqar, Mr. Essam Telmesani, Mr. Azhar Sayeed, Mr. Mueez Irfan, Mr. Javeed Nizami, Mr. Muhsin Siddiqui, Mr. Moh'd Sami and above all my honourable friend Sayyed Aiman Al-Maimani.

Finally, acknowledgement is due to King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, for fully supporting this research.

**Dedicated to my late uncle**

**Prof. A. Y. Aliyu**

**May Allah have mercy on him**

# Contents

# List of Figures

# List of Tables

# Abstract

**Name:** Mohammad Dikko S. Aliyu

**Title:** Efficient Algorithms for Collision Detection and Path Planning for Robotics Application

**Major Field:** Systems Engineering

**Date of Degree:** 1994

The problem of detecting collision between moving rigid objects in three dimensional space is considered. Two efficient algorithms that use linear programming technique are developed for solving this problem. The algorithms can detect exactly all possible collisions for objects moving on a general path in $\Re^3$ with simultaneous translation and rotation. Computational experience with the developed algorithms is also presented.

The potential field approach to the path planning problem is also considered. A new potential function that has the remarkable feature that it is free from any local minima in the free-space for any number of obstacles in the work-space is developed. Furthermore, a new approach to path optimization using an expanding sphere is also proposed. The new potential function is tested using a point mobile robot and smooth obstacles.

Master of Science Degree

King Fahd University of Petroleum and Minerals

Dhahran, Saudi Arabia

1994

# ملخص

الاسم : محمد ديكو عليو

عنوان الرسالة : خوارزميات لتحري التصادم وتخطيط المسار

الحقل الرئيسي : هندسة النظم

تاريخ الدرجة : ١٩٩٤م

في هذه الرسالة ندرس مسألة تحري التصادم بين الأجسام المتحركة في الفراغ ذي الثلاثة أبعاد بحيث تشمل الحركة الانتقال والدوران ، حيث تم تطوير خوارزميتين (طريقتي حل) تستخدمان البرمجة الخطية لحل هذه المسألة. وتستطيع هاتان الطريقتان اكتشاف جميع نقاط التصادم المحتملة للأجسام المتحركة في مسار عام في الفراغ. كذلك فإننا نورد نتائج تطبيق هاتين الطريقتين.

وبالإضافة الى ماسبق، فإننا ندرس أسلوب مجال الدالة من أجل تخطيط المسار، ونقدم دالة جديدة في هذا المجال تتميز بأنها خالية من النهايات الصغرى المحلية في الجزء من الفراغ الخالي من الأجسام، وكذلك نطور طريقة جديدة باستخدام كرة متمددة لمسألة التخطيط الأمثل للمسار، ونفحص هذه الطريقة الجديدة باستخدام روبوت ممثل بنقاط متحركة وأجسام إنسيابية.

**درجة ماجستير العلوم**

**جامعة الملك فهد للبترول والمعادن**

**الظهران، المملكة العربية السعودية**

**١٩٩٤م**

# NOTATIONS

| | |
|---|---|
| $\Re^n$ | Real Euclidean $n$-dimensional vector space. |
| $\| x \|$ | Euclidean norm of a vector $x$. If $x^T = (x_1, .....x_n)^T$, then $\| x \| = \sqrt{x_1^2 + ...x_n^2}$ |
| $\subset$ | Subset symbol |
| $\cap$ | Set intersection |
| $\cup$ | Set union |
| $\setminus$ | Set difference |
| $\mid$ | Such that symbol |
| $\{ . \}$ | Defines a set |
| $max$ | The maximum element of a set |
| $min$ | The minimum element of a set |
| $\| M \|$ | Determinant of a matrix $M$ |
| $\nabla$ | The differential operator |
| $Min$ | Minimization of a function |
| *subscipt* | An element of a vector or set index or matrix index |
| *superscript* | Matrix index |

# Chapter 1

# INTRODUCTION

The ultimate objective of robotics is to develop autonomous robots. These robots will have the capability to accept high-level descriptions of tasks and execute them without any further human intervention. The high level language will specify what the task is rather than how to do it. Applications of these robots in hazardous environments such as nuclear reactors, radiation zones and polluted areas; in milatary surveilance, space and undersea exploration; in automated industries such as AGV's and tele-operator systems; and in homes and hospitals, cannot be overemphasized.

The task of developing autonomous robots is however not an easy one. It is an undertaking with complex interrelationships between artificial intelligence, perception and control. It embraces many difficult problems amongst which **motion planning** is of central importance. Motion planning answers the following problem: How can a robot decide what motions to perform in order to achieve goal arrangement of physical objects ? This capability is absolutely necessary for an autonomous robot.

Generally speaking, motion planning involves such diverse aspects as computing collision-free

paths among possibly moving obstacles, coordinating the motions of several robots, planning sliding and pushing motions to achieve precise relations among objects, reasoning about uncertainty to build reliable sensory-based motion strategies, dealing with the models of physical properties such as mass, gravity and friction, and planning stable grasps of objects. Motion Planning also interacts with other major problems such as real-time motion control, sensing and task-level planning (see figure 1.1). Due to the complex interactions among these problems, the design of truly autonomous robots certainly requires concurrent solution of all these problems.

Three subsidiary problems to motion planning are: the **collision detection, obstacle avoidance** and **path planning** problems. The collision detection problem is to determine whether a known robot configuration (or path) would cause collision between potentially colliding parts such as links, payloads or obstacles in the three dimensional workspace. While the obstacle avoidance problem is to modify a known robot path so as to avoid foreseen or unforeseen obstacles. Strictly speaking, obstacle avoidance is a control strategy, and involves the determination of appropriate forces/torques to drive the robot onto a collision-free path [10,36,45,50,68,85]. This may further require the optimization of a certain criteria (eg. time, distance or energy) in the presence of the constraints on the path, torque, end-effector, friction etc. However, an easier problem involves the determination of a purely geometric path which is collision-free, without regard to the forces required to move the robot along this path. This is known as the path planning (or find-path) problem. These three subproblems are part of the paraphernalia of autonomous robot motion planning. The integration of these subsystems with other subsystems such as trajectory planning and trajectory tracking, achieves in a more simplified manner, the solution of the generalized motion planning and control problem.

The interaction of the various sub-systems can be explained in the light of Fig.1.1. From

a task planner[27] we obtain an ordered sequence of actions to be executed to perform a task. These actions are decomposed into robot motion commands which can be implemented in a standard robot programming language. The motion commands involve point-to-point motions, transfer movements, sliding and pushing, gripping and grasping etc. The selection of appropriate geometric path (eg. collision-free) to achieve the desired motion is done by the path planner. The output of the path planner is therefore an ordered sequence of points in cartesian space (task space) which represent a collision-free path if we connect them properly (eg. by straight line segments or splines). The trajectory planner in-turn accepts these input variables including path constraints, and generate a sequence of time-based intermediate configurations of the robot (position and orientation, velocity and acceleration), expressed usually in joint space, from initial location to the final location. These are then fed to the robot controller or trajectory tracker which computes the forces/torques to be exerted by the actuators at each time in order to perform the desired motion.

The above discussion has focussed on motion planning for general robotic systems without descrimination between mobile and manipulator robots. However, there is a fundamental difference! In the context of mobile robots, path planning is simplified by restricting the robot to three degrees of freedom on the plane, and motion planning is referred to as **navigation** [69]. Nevertheless, much of the work in path planning for mobile robots is derived from earlier work in path planning for manipulators.

In this thesis, we are interested in developing efficient algorithms for solving the collision detection and path planning problems. In both problems, we are interested in developing off-line schemes in which there is accurate knowledge of the nature, position, orientation and motions of the obstacles. For collision detection, we will consider a general approach to the problem, while

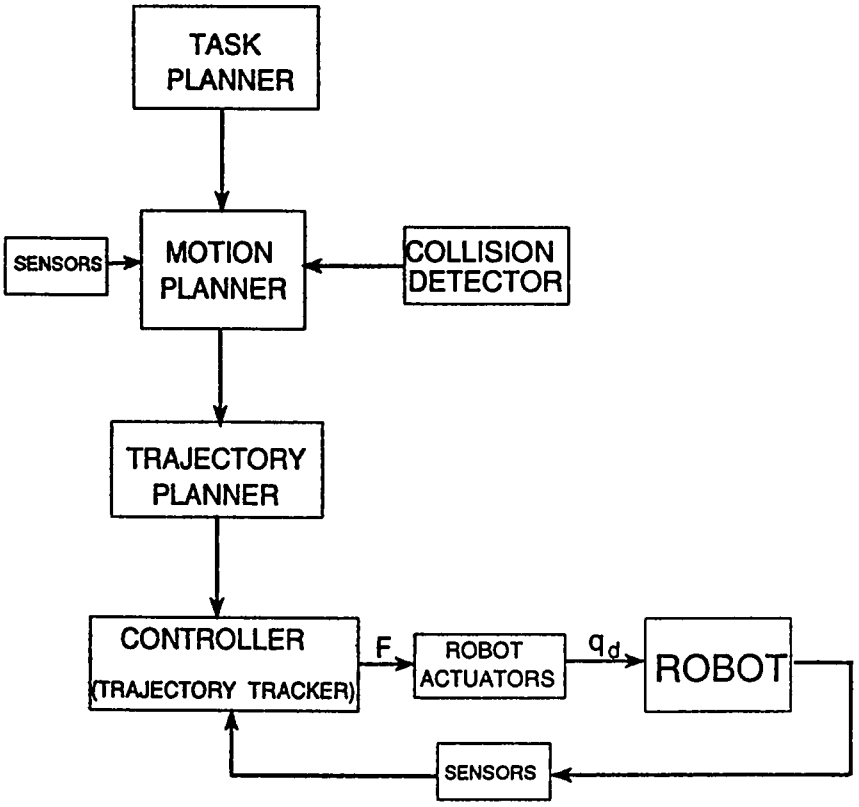


Figure 1.1: Heirachical Control of an Autonomous Robot

for path planning, we will consider only local approaches. In particular, we shall consider the **potential field** approach to path planning.

The organization of the thesis will be as follows: In the remaining portion of chapter 1, we will give formal definitions of the problems, introduce some terminology and discuss various methods of solid modelling schemes which will form the basis for our algorithms. At the end of the chapter, we shall define our research objectives. In chapter 2, we will give detailed literature review of what has already been done on the two problems of collision detection and path planning. This will give a clear picture of our aims and objectives in this thesis, and give us direction on how to achieve them. In chapter 3, we will develop and present our collision detection algorithms.

This will be followed by a discussion on the computational experience on the collision detection algorithms in chapter 4. In chapter 5, we will demonstrate how the collision detection algorithms can be used in a robotic application. This will be followed in chapter 6 by our new approach to the potential field method for path planning. We will develop the new approach and give computational experience on it. Finally, in chapter 7, we will give a summary and conclusions on the whole thesis, and give recommendations for future work in the area.

## 1.1 Problem Definition

In this section, we give general definitions of the motion planning and collision detection problems and show the relationship between the two. Then we will review the literature on the two problems.

**Basic Motion Planning Problem**

Let $A$ be a single rigid object -the robot- moving in a workspace $W \subset \Re^n$ ($n = 2$ *or* 3), the Euclidean space.

Let

$B_1, B_2, .....B_N$ be fixed rigid objects distributed in $W$, called obstacles.

Assume that both the geometry of $A$, $B_1, B_2, ...B_N$ and the locations of the $B_i's$ in $W$ are accurately known. Assume further that no kinematic constraints limit the motions of $A$ (we say $A$ is a free-flying object).

The problem is : Given an initial position and orientation, and a goal position and orientation of $A$ in $W$, generate a path $\Gamma$ in $W$ specifying a continuous sequence of positions and orientations of $A$ avoiding contact with the $B_i's$, starting at the initial position and orientation, and terminating at the goal position and orientation. Report failure if no such path exists.

Defined in this way, the basic motion planning problem with purely geometric path searching is usually referred to as **path planning.**

**Collision Detection Problem**

A more general definition of collision detection may be stated as follows:

Given representations of $N+1$ objects $A$, $B_1, B_2, ....B_N$, whose locations in space at any time $t$ are given by functions $L(t)$ ,$L_1(t)$ ,$L_2(t)$,....$L_N(t)$ , respectively, over a time interval $[t_s, t_f]$, determine whether any pair of the objects occupy some common space at the same time during this interval [1].

In particular, check whether

$$\bar{A}(t) \cap \bar{B}_i(t) \neq \emptyset \; for \; t \in [t_s, t_f] \; , i = 1, 2....N$$

where $\bar{A}(t)$, $\bar{B}(t)$ are the sets representing the two objects $A$ and $B_i$ at any time $t$.

In the above definition, $A$ is typically a robot moving in a workspace $W$ in which there are

obstacles $B_i, i = 1, 2...N$. Furthermore, the connection between the motion planning problem and the collision detection problem can easily be made out from the above definitions. The collision detection problem is the path planning problem with a hypothesized path. The path is tested for any collisions, if it fails, another is hypothesized until a collision-free path is found. This is one application of collision detection and is known as the propose-and-correct approach to path planning.

We are especially interested in solving the collision detection problem for robotics, but the solution has application in other problem areas such as VLSI and electronic circuit layout, cloth cutting, bin packing, assembly planning and geometrical modelling and simulation [16].

In robotics, the collision detection problem is of interest because of the following reasons:

- Off-line robot programming systems are mostly based on interactive graphics under human control. A fast collision detection facility can greatly assist the programmer in spotting collisions which would otherwise be very difficult.

- A collision detector may be used to test for a valid path in a propose-and-correct approach to path planning (figure.1.1).

- Present collision avoidance schemes are complicated and can handle only a limited range of shapes and motions.

## 1.2 Requirements for Collision Detection

In order to describe collisions precisely, it is useful to define three possible interactions between a pair of objects, viz: separation, contact and interference. Two objects are said to be "separated"

if their intersection as sets of points is empty, "contacting" if their intersection includes only points from the surfaces of the objects, and "interfering" when the intersection includes points from the interior of either object. In most cases, collision refers to contact brought about by relative motion between the objects. An algorithm for collision detection should satisfy the following requirements:

1. No detection miss.

2. Locate exact time and point of the earliest collision.

3. Accurately approximate the motions of the objects.

4. Exclude all false collision report.

## 1.3 Terminology

Often, we will be mentioning some terms in the literature and in our discussion, hence, it is appropriate to introduce them at this point.

**Configuration Space:** A configuration $q$ of $A$ is a specification of the position $'P'$ and orientation $\Theta$ of the reference frame of $'A'$, $F_A$, with respect to a world reference frame $F_W$. The *configuration space* of $A$ is the space $C$ of all the configurations of $A$.

**Path:** The path of $A$ from the configuration $q_{init}$ to the configuration $q_{goal}$ is a continous map :

$$\Gamma : [0,1] \rightarrow C$$

with

$$\Gamma(0) = q_{init} \quad , and \quad \Gamma(1) = q_{goal}$$

$q_{init}$ and $q_{goal}$ are the initial and goal configurations of the path, respectively.

**C-Obstacle:** Every obstacle $B_i, i = 1,2....N$ in the workspace $W$ maps in $C$ to a region

$$CB_i = \{q \in C \mid A(q) \cap B_i \neq \emptyset\}$$

which is called a *C-obstacle.* The union of all the C-obstacles $\bigcup_{i=1}^{N} CB_i$ is called the *C-obstacle region*, and the set

$$C_{free} = C \setminus \bigcup_{i=1}^{N} CB_i = \{q \in C \mid A(q) \cap (\cup_{i=1}^{N} B_i) = \emptyset\}$$

is called the *free space.* Any configuration in $C_{free}$ is called *free configuration.*

## 1.4 Representation of Objects

In this section, we will discuss some of the methods used in modelling solid objects in three dimensional space. We shall discuss six methods, out of which we will choose the one most appropriate for our work. The method of representation of the objects is very important because it characterizes the nature of the algorithms and their complexity. These methods are: 1) Sweep representation; 2) Constructive solid geometry; 3) Boundary representation; 4) Cell decomposition; 5) Spatial occupancy enumeration; and 6)Octree representations.

### 1.4.1 Sweep Representation

This is a description of a surface and the volume it generates when it sweeps along a trajectory in curvilinear coordinates. Techniques in analytical geometry are used to derive the equations of these surfaces. They are very well structured but the generality of these schemes are not well understood. However, elegant two dimensional representations have been demonstrated. Three

dimensional applications are restricted to simple cases, and the scheme is of primary interest for high level descriptions of objects naturally decomposable into elongated elements.

### 1.4.2 Constructive Solid Geometry

These are expressions of primitive portion of space and combinatorial (intersection, union, ....) and motional (translate,rotate) operators. Varieties of possible choices of primitive portion of space are possible, such as bounded primitive solids and unbounded half-spaces. These schemes are extensively used in manufacturing systems

### 1.4.3 Boundary Representations

These are mostly used in computer graphics and are the most familiar. Solids are represented by their bounding faces which inturn are represented by their bounding edges, which inturn are bounded by vertices. They can only represent polyhedra adequately and other solids have to be approximated as polyhedra with many faces.

#### Representation of Polyhedral objects

Convex polyhedral objects can be modelled using boundary representation in $\Re^n$, $(n = 2, 3)$ by polyhedral sets defined by the intersection of finite hyperplanes or as convex hulls of finite number of vertices. Both representations have their advantages, and one will choose one of the two depending on the application. In robotics, both representations are useful as we are going to see.

In the first representation, the objects are defined by a system of linear inequalities of the

form:

$$\Lambda_i x \leq b_i \ ; i = 1, 2....N \tag{1.1}$$

Where $N$ is the number of objects, $x \in \Re^n$ ,$\Lambda_i$ is $(m_i \times n_i)$, and $b_i$ is $(n_i \times 1)$. The above system of linear inequalities represents an intersection of the halfspaces whose bounding hyperplanes define the faces of the object. This representation can be obtained from a complex object( polyhedron) by projecting the planes of the faces of the object onto the three axes of the reference frame and determining their intercepts. The accuracy of this representation depends on the number of hyperplanes used; the more the number of hyperplanes used, the better the accuracy.

In the second representation, the objects are represented by point sets defined as convex hulls of a finite number of vertices.

$$co\mathbf{X} = \{x = \sum_{j=1}^{l} \lambda_j x_j : \sum_{j=1}^{l} \lambda_j = 1 \ ; \lambda_j \geq 0 \, ; j = 1...l\} \tag{1.2}$$

where $\lambda_j \in \Re$ and $x_j$ are the vertices of the object. This representation implies that any point on the object is represented as a linear combination of the vertices of the object with nonnegative coefficients. By the Caratheodory theorem[18] $l \geq n + 1$.

In the above representations, smooth objects cannot be represented adequately, they can only be approximated. However, smooth compact objects (eg. spherical or ellipsoidal) can be represented using analytic inequalities of the form:

$$g(x) \leq 0 \, ; x \in \Re^n \, ; g(x) \subset C^3[0, \infty) \tag{1.3}$$

Nonconvex objects can also be represented as union of simple polyhedral sets.

### 1.4.4 Cell Decomposition

These are generated by triangulation, and a model is decomposed into elementary solids (usually tetrahedra) meeting exactly at a common face, edge or vertex. They are suitable for computing certain topological properties of represented solids and are extensively used in finite element analysis. They are also the basis for certain robot motion planning algorithms[53].

### 1.4.5 Spatial Occupancy Enumeration

These are data structures of voxels (volume elements), usually cubes lying on a square grid, of the solid. They are simple but, lead to large amounts of storage. They can be considered as special cases of cell decomposition in which the cells are of identical size and shape.

### 1.4.6 Octree Representations

These are special cases of spatial occupancy enumeration schemes. An octree is a hierachical data-structure aiming at reducing the amount of redundancy in spatial occupancy enumeration schemes. A cubic reference portion of three dimensinal space is divided into eight octants. Each octant can further be recursively decomposed into smaller octants leading to a tree of order eight. The nodes of the tree are labelled according to their position on the solid i.e. whether empty, mixed or full (see Fig.1.2) Octrees are widely used in robot motion planning , computer graphics, computer vision, CAD, e.t.c.

## 1.5 Research Objectives

The following are our research objectives in this thesis:

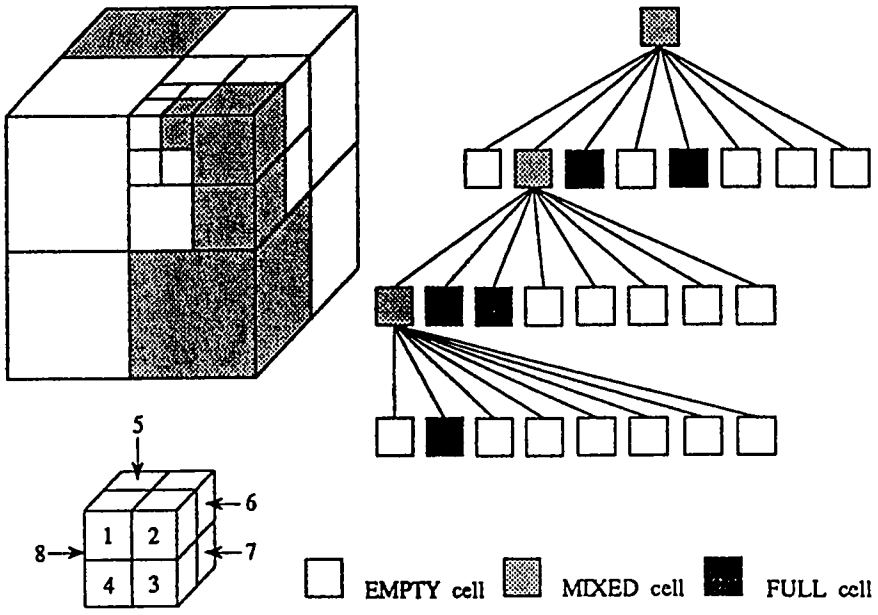


Figure 1.2: Octree decomposition of a solid block(adopted from [53])

1. To develop efficient algorithms for collision detection using different representations of objects.

2. To introduce a new potential field approach to path planning and obstacle avoidance.

3. As a minor objective, we shall also make an extensive literature survey of collision detection and path planning algorithms in robotics.

# Chapter 2

# LITERATURE REVIEW

## 2.1 Introduction

In this chapter, we will give a comprehensive literature review of the collision detection and path planning problems. These two problems have received substantial attention from researchers and the literature is replete with various contributions. To make our presentation neat and elaborate, we will classify the various algorithms for collision detection and path planning into five classes each. These classifications will be based on the methods of representation of the objects and the nature of the algorithms. For the collision detection problem, the five major classes of algorithms are: a)Swept volume approach; b) Intersection calculation; c)Minimum distance calculation; d)Multiple interference detection; and e) Constructive solid geometrical schemes. However, we will first give a detailed presentation of all the algorithms in their own perspective before attempting to classify them. For path planning on the other hand, we will first give the classification of the algorithms since this already exists, and then briefly highlight their nature. However, in the case of the potential field approach which we will be extending in

this thesis, we will give a full literature review of what has been done so far.

## 2.2 Collision Detection

One of the earliest work in collision detection (interference checking) was presented by Boyse[11]. He treated separately the cases of uniform translation and rotation around a fixed axis. The algorithm involves a brute force approach in which each edge of the moving object is intersected with each face of the stationary object. The algorithm is so unrefined that it is only applicable to simple objects and simple motions.

Ahuja et al. [18] have described two methods for detecting intersections among three dimensional objects. The first method involves detecting overlap among the projections of the objects on a given set of planes, while the second method uses a three dimensional octree representation (chapter 1) of the objects. Interferences are detected by traversing the octrees of the objects in parallel. If there exists at least a pair of corresponding nodes on the octrees of two objects having opposite labels, then there is interference between them. However, these methods do not give an exact method for interference detection and cannot be applied to the case of complex motions. THis is because, the projections of the objects on a given set of planes is difficult to obtain and the representation of complex motions using octrees is computationally intensive.

Shigematsu et al.[78] have described an algorithm based on the simplex method. The objects are represented as convex polyhedra defined by the intersection of halfspaces. The intersection of the union of the half spaces from two objects gives the interference region if there is any. The algorithm is attractive but it is limited to only static objects.

Cameron[15,16] has described three methods for the "clash detection" problem for moving

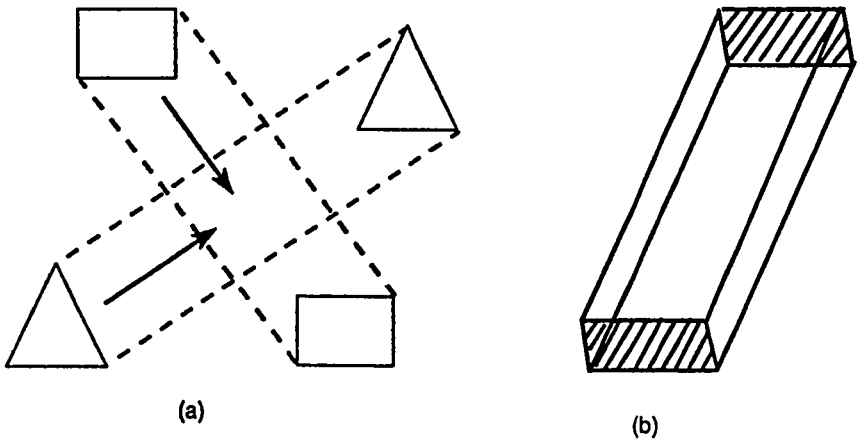


Figure 2.1: (a)Sweeping and (b)Extrusion in two dimensions

polyhedra viz : multiple interference detection which is based on sampling the motion at discrete points of time and doing static interference detection at each point; four dimensional intersection detection which is based on the representation of the objects in a four dimensional space (with time as the fourth); and sweeping which is based on computing the swept volume generated by the objects along their trajectories (see Fig.2.1) None of these methods compute exact collision points for paths with simultaneous translation and rotation.

Hayward [37] has also described an algorithm based on octree representation of the robot and obstacles, or alternatively, the free-space of the robot. Collision detection for a point on the moving object (the robot) is determined by inspection of the octree nodes to characterize the point by the label on the octant to which it belongs; whether it is labelled 'empty', 'full', or 'mixed' (see Fig.1.2). The algorithm has several drawbacks: First, is the amount of storage required for the representation. Second, transformations on octrees such as translations, rotations, erosions, growth e.t.c. are computationally intensive. Thirdly, the algorithm is combinatorial in the features of the objects.

Culley and Kempf [23] have described an algorithm based on velocity and distance bounds. The algorithm utilizes the maximum velocities of the objects to determine the minimum time it will take the objects to reach contact. The shortest distance between the objects is also computed to determine their contact or separation. However, the algorithm does not compute the exact collision points and cannot be readily implemented.

Meyer[65] has described an algorithm using the shortest distance between the objects (represented as boxes). For complex objects, this distance is very difficult to compute [32]. Furthermore, the algorithm does not treat the case of simultaneous translation and rotation, and does not compute exact collision points.

The first algorithm that computes exact collision points with simultaneous translation and rotation for a moving object amidst obstacles was described by Canny[17,18]. He used quaternions to describe the orientation of the objects and then considered three types of interactions between the moving object $A$ and the obstacle(s) $B$. *Type-A* contact occurs when a vertex of $B$ touches a face of $A$; *type-B* contact occurs when a vertex of $A$ touches a face of $B$ ; and *type-C* contact occurs when an edge of $A$ touches an edge of B (Fig.2.2). These three interactions cover all the possible types of contacts between the objects. The basic algorithm is based on the following constraint equations.

**For type A contact**

$$[\bar{Q}\bar{F}_A\bar{Q}^*(1 + \vec{p}_b - \vec{x})] = 0 \tag{2.1}$$

Where $A$ is moving and $B$ is stationary. The "bar" denotes quaternion while the arrow denotes vector.

$\bar{Q} = q_0 + \vec{q}$ defines the orientation of A

$\vec{x}$ is the translation vector of A.

$\vec{p}_B$ is the position vector of any vertex of B.

$\bar{F}_A = d_A + \vec{n}_A$ defines a face of A.

$\vec{n}_A$ is the unit outward normal of the face $\bar{F}_A$

$d_A$ is the normal distance from the origin to the face $\bar{F}_A$

**For type B contact**

$$[\bar{F}_A(1 + \bar{Q}\vec{p}_A\bar{Q}^* + \vec{x})] = 0 \tag{2.2}$$

where

$\bar{F}_B = d_B + \vec{n}_B$ defines a face of B.

$\vec{p}_A$ is the position vector of any vertex of A.

**For type C contact**

$$[(\bar{Q}\vec{p}_A\bar{Q}^* + \vec{x} - \vec{p}_B)\bar{Q}\vec{e}_A\bar{Q}^*\vec{e}_B] = 0 \tag{2.3}$$

Where $\vec{e}_A$ and $\vec{e}_B$ represent unit direction vectors of the edges of $A$ and $B$ respectively. The square brackets imply the scalar part of the vector quaternion product.

$\bar{Q}$ and $\vec{x}$ are typically functions of a parameter "$s$" that defines the path. Substituting these in equations (2.1), (2.2), and (2.3), results in cubic algebraic equations in $s$. These are solved to determine the value of $s$ for which there is a contact, and subsequently determine the position and orientation of the objects.

The only shortcoming of this algorithm is in defining the faces, vertices and edges of the

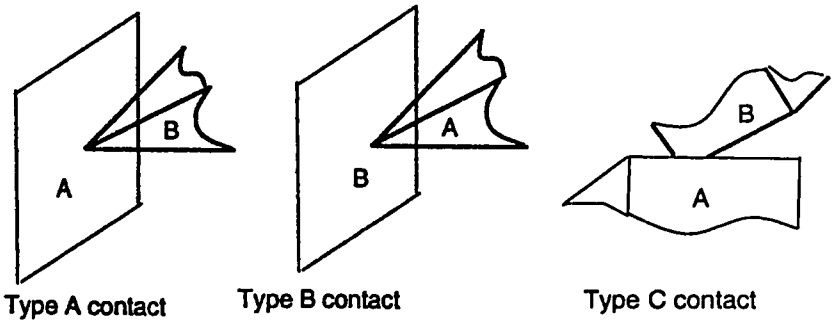


Figure 2.2: Three types of interactions between colliding objects

objects; especially, when the objects are not simple (e.g. smooth objects ). Furthermore, the algorithm has to test for each type of contact for all vertices, faces and edges. This implies long computational time and high cost.

Lozano-Perez[58] has given expressions for contact angles for manipulator links and obstacles in joint space. These are useful for motion planning.

Basta et al [5] have presented an algorithm for planning collision-free motion of two robot arms in a common workspace. The algorithm uses a sphere model for the wrists and detects collisions along straight-line segments of the paths of the two robots. If at any point the distance between the two segments is less than the sum of the radii of the two spheres representing the two wrists, then there is a potential collision. The set of points for which the above applies gives a Parametric-Space-Potential-Collision Region Diagram (PSPCRD). Using trajectory information, the time range when the potential collisions along each path occur can be determined. Any overlap in the two time ranges suggests, but does not guarantee, the existence of a space-time collision.

Another exact algorithm developed by Kawabe et al.[44] also considers three types of interactions between the objects: *Type 1 contact* (a vertex of a moving object touches a face of a

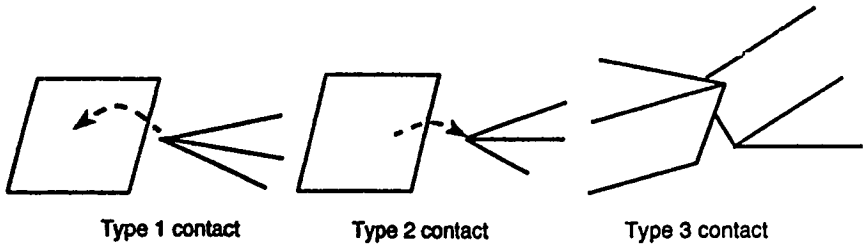


Figure 2.3: Interactions between objects features

stationary object); *type 2 contact*(a face of a moving object touches a vertex of a stationary object); and *type 3 contact* (an edge of a moving object touches an edge of a stationary object )( see Fig.2.3). The detailed algorithm is based on the following three constraints equations.

**For type 1 contact**

$$aX_k(t) + bY_k(t) + cZ_k(t) + d = 0 \tag{2.4}$$

Where $(X_k(t), Y_k(t), Z_k(t)) = T_k(t)\vec{P}$ are the coordinates of the moving vertex $P$, $T_k(t)$ is the homogeneous transformation matrix defining the relative motion of the object $i$ with respect to object $j$ and $a, b, c, d$ are constants of the plane of the face.

$X_k(t), Y_k(t), Z_k(t)$ are typically cubic functions of time, hence, the degree of equation (2.4) is three. This gives three answers, and the minimum that satisfies the bounds in time and object dimensions is selected.

**For type 2 contact**

$$a(t)X + b(t)Y + c(t)Z + d(t) = 0 \tag{2.5}$$

Where $X, Y, Z$ are the coordinates of the stationary vertex and $a(t), b(t), c(t), d(t)$ are cubic fun-

tions of time. The order of eqn.(2.5) is three,and we get three answers out of which we select the best.

**For type 3 contact**

$$a(t)u + b(t) = cv + d \tag{2.6}$$

or

$$[a(t) \;\; -c \;\; b(t) - d] \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \tag{2.7}$$

or

$$|a(t) \;\; -c \;\; b(t) - d| = 0 \tag{2.8}$$

Where $u$ and $v$ are parameters, $c$ and $d$ are constant vectors , and $a(t)$ and $b(t)$ are vectors each of whose elements is a cubic function of time.

The degree of eqn.(2.8) is six, and we obtain six answers out of which we select the best.

Finally, for the three types of contacts, we select the minimum answer as the time when collision occurs for the first time during the motion.

The drawback of this algorithm is in the accurate determination of the functions $X_k(t), Y_k(t), Z_k(t), a(t$ and $d(t)$ for the various faces, edges and vertices of the objects. Secondly, the computational cost of the algorithm is not low because of its combinatorial nature.

Bonner and Kelley [9] have presented an algorithm based on a successive spherical approximation (SSA) of the objects. The SSA representation of an object is made up of a hierachy of representation levels based on the division of an object-encircling sphere into spherical sectors. The most coarse approximation is a pair of spheres with a common centre. The outer sphere

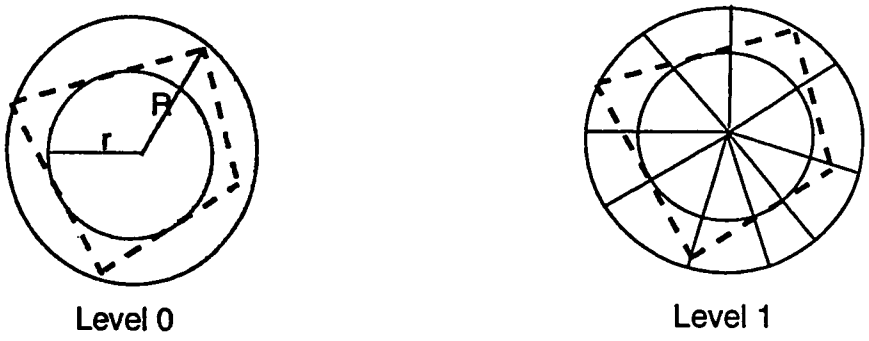


Figure 2.4: SSA representation of a convex planar object

contains the object and the inner sphere is contained within the object. The next finer approximation is formed by dividing the spheres into spherical sectors defined by the object faces (see Fig.2.4). Collision detection is done by incrementally checking for intersection between these spherical sectors. Again, the algorithm makes gross approximations and can only treat limited types of motions.

A relatively recent work on the collision detection problem has been reported by Gilbert and Hong [34]. The algorithm applies to convex objects and uses, as a substep, the computation of the shortest distance between the objects [32,33]. A brief outline of the basic algorithm is given here.

Let the spaces occupied by the potentially colliding objects be represented by point sets $K_1, K_2 \in \Re^n$ $n = 2$ *or* 3. Thus,

$$K_i(q) \stackrel{\text{def}}{=} \{x = R_i(q)w + p_i(q) : w \in C_i\};\ i = 1,2 \tag{2.9}$$

Where $q$ is the configuration vector which belongs to a configuration space $Q$, $p_i(q)$ is the translation vector, $R_i(q)$ is the $n \times n$(orthogonal) rotation matrix, $C_i \subset \Re^n$ is the set of points which describes the space occupied by object $i$ in its reference position and orientation. The configura-

tion vector moves along a path in $Q$ which is specified parametrically by a continuous function $q : \Theta \rightarrow Q$. $\Theta = [\theta_s, \theta_f]$, where $q(\theta_s)$ is the starting point on the path and $q(\theta_f)$ is the final or ending point on the path.

Define the support functions of the sets $h_{K_i} : \Re^n \rightarrow \Re$ by

$$h_{K_i(\theta)}(\xi) \stackrel{\text{def}}{=} max\{\langle \xi, z_{ij}(\theta) \rangle : j = 1, ...M_i\}; \; i = 1, 2 \tag{2.10}$$

where $\xi$ is the normal to the supporting hyperplane of the set, and $M_i$ are the number of vertices of the object. Also, define $d_k$ as the shortest distance between the two objects, $\xi_k = \nu_1 - \nu_2$ and $\nu_1, \nu_2$ are a pair of nearest points [4,91] on the two objects respectively ( see Fig.2.5). The collision detection problem (CDP) is then stated as follows :

Assume

$$K_1(\theta_s) \cap K_2(\theta_s) = \emptyset$$

,

find the collision point

$$\theta^* = min\{\theta \in \Theta : K_1(\theta) \cap K_2(\theta) \neq \emptyset\}.$$

or show that

$$K_1(\theta) \cap K_2(\theta) = \emptyset \; for \; all \; \theta \in \Theta$$

The basic collision detection algorithm is as follows.

*Basic Collision Detection Algorithm (BCDA)*

*step 1* set k=0 and $\theta_0 = \theta_s$

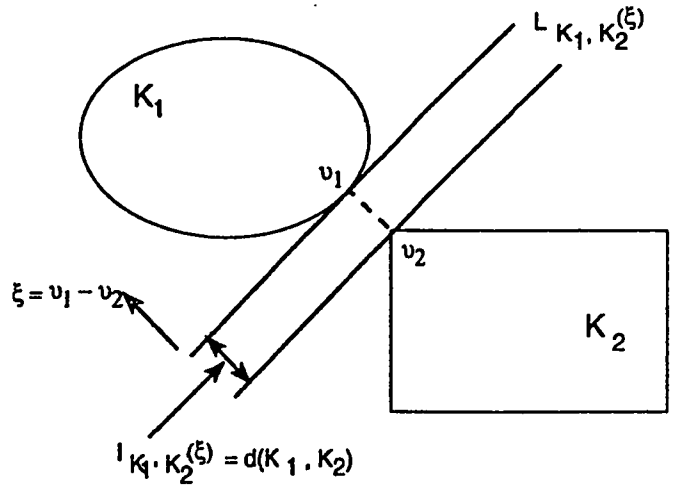


Figure 2.5: The separating slab generated by the nearest points $\nu_1$ and $\nu_2$

*step 2.* Determine $d_k$ *and* $\xi_k$.

*step 3.* If $d_k \leq \epsilon$, stop and set $\theta^* = \theta_k$

*step 4.* solve, if possible, the problem of finding the smallest root of $l(\theta; \xi_k) = 0$ *on* $[\theta_k, \theta_f]$.

Equivalently, find, if it exists, the smallest root of

$$F_{k(\theta)} = h_{K_1(\theta)}(-\xi_k) + h_{K_2(\theta)}(\xi_k) = 0$$

*step 5.* If the root finding problem has no solution, stop. There is no collision on $\Theta$

*step 6.* Let $\theta_{k+1}$ be the solution of the root finding problem. Increment k by 1 and go to *step 2.*

End

The main computational effort in this algorithm is in the determination of $d_k$ and the root finding problem which corresponds to a point on the path where the distance between the objects is zero . These two subproblems are very difficult[32,34] and contribute to the inattractiveness

of the algorithm.

Two more recent works on the collision detection problem have been reported by Schweikard [76] and Kyriakopoulos et al.[51]. Schweikard has described a polynomial time collision detection scheme for manipulator paths specified by joint motions. But obstacle motion is normally described in task space. Kyriakopoulos and Saridas have developed an on-line algorithm for distance estimation and collision prediction. The descriptions of the objects and their motion are uncertain, and have to be estimated by sensing and filtering.

Again Kyriakopoulos et al. [52] have given a swept volume approach for on-line collision prediction of a mobile robot and mobile obstacles. Consider a convex polyhedral description of the mobile robot and a moving obstacle represented by

$$\Lambda_r x_r \leq B_r \tag{2.11}$$

$$\Lambda_o x_o \leq B_o \tag{2.12}$$

respectively. Where $x_r, x_o \in \Re^2$. $A_r, B_r$ are functions of the path parameter $s$, while $A_o, B_o$ are functions of time $t$. The trajectories of the mobile robot and the moving obstacle under horizontal translation can be parameterized by

$$x_r(s) = x_r + \lambda v_r, \tag{2.13}$$

$$x_o(t) = x_o + \mu v_o \tag{2.14}$$

which on substitution into equations (2.11) and (2.12) give the swept areas of the objects respectively. $v_r$ and $v_o$ are the robot and obstacle velocities respectively. If their swept areas intersect, then

$$x_r + \lambda v_r = x_o + \mu v_o \tag{2.15}$$

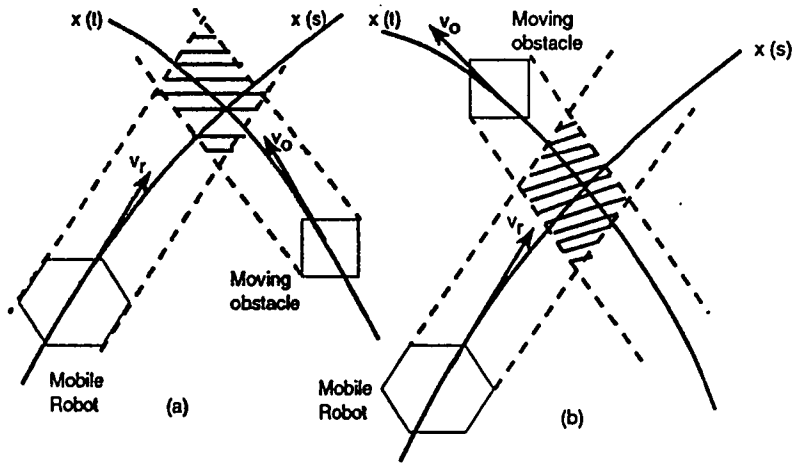


Figure 2.6: (a)Alarming configuration (b)Non-alarming configuration

Introducing equation (2.15) into equations (2.11) and (2.12), their area of intersection is given by the set

$$S = \{X \in \Re^6 \mid AX \leq B\}, \tag{2.16}$$

where

$$A = \begin{pmatrix} A_r & 0 & A_r v_o & -A_r v_r \\ 0 & A_o & -A_o v_o & A_o v_r \end{pmatrix}$$

$$X = \begin{pmatrix} x_o \\ x_r \\ \mu \\ \lambda \end{pmatrix}$$

$$B = \begin{pmatrix} B_r \\ B_o \end{pmatrix}$$

The solution of the above system can be obtained using linear programming[3,7,25]. However, a collision will occur (alarming configuration in Fig.2.6) iff;

$$\lambda_{max} = max_{X \in S}\ \lambda > 0,$$

and

$$\mu_{max} = max_{X \in S}\ \mu > 0.$$

Gallerini R. et al. [29] have described implementation of algorithms based on linear programming. The algorithms use similar ideas in [66,78] with practical considerations. The algorithms treat the case of linear traslations only.

In general, the existing algorithms for collision detection can be divided into five categories :

**a) Multiple Interference Detection:**

In this class of algorithms, the position and orientation of the moving object is calculated at a reasonably sampled time and interference check is performed at each point. Methods for detecting interferences among stationary objects are therefore utilized [20,70]. The time interval between successive checks is kept short to avoid a collision miss. However, if this time is made too short, the computation time may become too long for practical purposes. The algorithms in [15,23,87] belong to this category.

**b) Swept Volume:**

In this class of algorithms, the volumes swept out by the objects over their motions are computed and collisions are declared if these swept volumes intersect. But at present, it is difficult to obtain a swept volume unless the motion of the objects is described by linear or quadratic equations. Besides, an intersection of the swept volumes may not result in a collision if the objects are moving. Furthermore, even if a collision is detected, it may be very difficult to obtain the exact time and position of the collision. Two of the algorithms in [15] and the algorithms in

[1,11,16,52] belong to this category.

**c) Intersection Calculation:**

In this class of algorithms, the trajectories of the faces, edges and vertices of an object are described by functions of time, and the conditions for contact are used to derive the equations for collision. By solving these equations, the exact time and position of the collisions are obtained. The algorithms in [11,17,18,44,58,76] belong to this class.

**d) Minimum Distance Calculation:**

This class of algorithms compute the minimum distance between the objects and determine the point on the path when this distance will shrink to zero. The distance is usually computed at discrete times along the path. The algorithms in [5,9,20,23,29,34,62,51,65,74,76] belong to this class.

**e) Constructive Solid Geometry Schemes:**

This class of algorithms use data structures in the form of octrees/quadtrees [54] to decompose the objects and the free-space. Collisions are determined by inspection of the labels on the nodes of the octrees representing the objects. The algorithms in [2,37,84] belong to this class.

## 2.3 Path Planning

Path planning has emerged as one of the most challenging problems in the development of autonomous robots. Consequently, a lot of research efforts have been devoted to it over the past two decades. This literature is very rich, and we will not try to exhaust it. However, we will

mention the most popular approaches that have been developed and concentrate on the potential field approach which we hope to extend in this thesis.

Over the years, five main approaches to the path planning (or path finding) [12,14,57] problem have emerged. These are :

- Roadmap methods.
- Cell Decomposition methods.
- Optimal Control methods.
- Mixed methods.
- Potential Field methods.

Furthermore, algorithms can be classified as being exact or heuristic. Exact algorithms either find a solution or prove that none exists, and they tend to have high complexity. Heuristic methods on the other hand, attain faster solutions at the expense of reliability.

The roadmap method consists of constructing a network of one dimensional curves that capture the connectivity of the robots free-space ($C_{free}$) or its closure. Once a roadmap $\mathcal{R}$ has been constructed, path planning reduces to connecting the initial and the goal configurations to points in $\mathcal{R}$ and searching $\mathcal{R}$ for a path between these points. Variants of the roadmap method include *visibility graph, freeway net, and silhouette* [13,14,18,56,58]. Fig.2.7 shows a visibility graph with polyhedral obstacles.

Cell decomposition involves the decomposition of the robot's free-space into simple regions called cells such that a path between any two configurations in a cell can be easily generated. The graph arising from connecting adjacent cells is called the *'connectivity graph'*. Two major

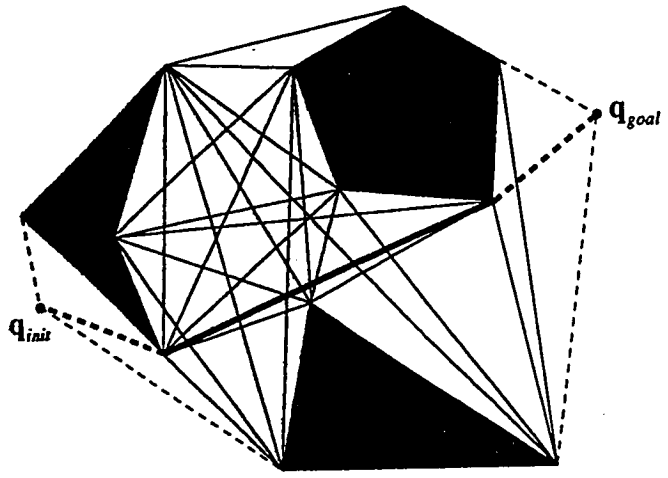


Figure 2.7: Visibility graph with polygonal obstacles (adopted from [13]).

types of cell decomposition are the exact and approximate cell decomposition (see Fig.2.8and references [28,35,43,55,59,76]).

Optimal control methods for path planning determine a path or trajectory while minimizing certain performance indexes - usually time or distance. The path is parameterized as a function of a scalar variable and the objective function (or performance index) is minimized subject to constraints due to the robot dynamics and kinematics. In this way, the derived path is both dynamically and kinematically optimal. The path is also smooth. An added advantage to this method is that, it can directly yield the optimal controls required to move on the path. However, there is a penalty to pay. They are computationally intensive [8,31,40-42,46,49,60,61]-[66,73,77,79,80-83].

A formulation of the optimal path planning algorithm due to Gilbert and Johnson [31,41,42,] is outlined below:

The problem is to minimize

$$J = \int_0^T dt$$

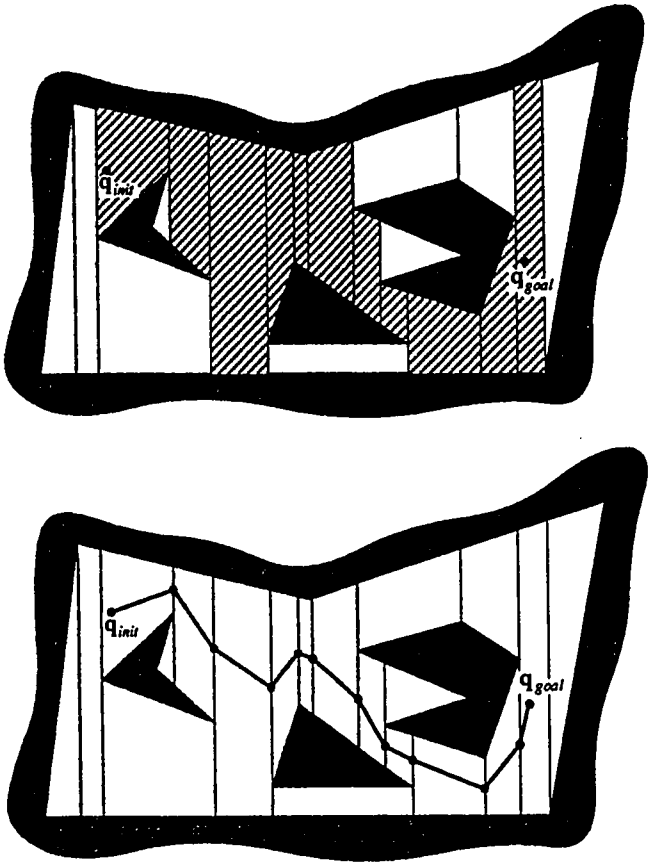


Figure 2.8: A path in the connectivity graph from exact cell decomposition (adopted from [53]).

s.t.

$$\dot{q} = (M(q))^{-1}(H(q)u - F(q,\dot{q}))$$

$$h(q(0), q(\tau)) = 0$$

$$g(q(t), u(t)) \leq 0$$

$$d_{ij}^{\circ} - d_{ij}(q(t)) \leq 0$$

$$t \in [0,\tau];\ \ i,j \in I \subset \{1,2..N\}$$

$$0 < \tau < a$$

where

$$d_{ij}(q) = min\{|z_i - z_j| : z_i \in K_i(q),\ z_j \in K_j(q)\}$$

and

$$K_i(q) = T_i(q)C_i + p_i(q)$$

is as defined in section 2.2 above, $\tau$ is either free or fixed, $a$ is fixed, $q \subset C^n[0,a]$ ,$u \in U$ (set of measurable bounded functions from $[0,a]$ to $\Re^p$) is the actuator input. $M(q) \in \Re^{p\times p}$ is a nonsingular inertia matrix, $F(q,\dot{q})$ represents a variety of force terms including actuator damping, $H(q) \in \Re^{p\times p}$ is a nonsingular actuator coupling matrix. $h$ defines the initial and terminal states, $g$ defines the states and control constraints, and $d_{ij}^{\circ}$ is a margin for error in avoiding collision.

The mixed methods combine various techniques such as cell decomposition, roadmap, orthogonal projections, collision checking, shortest-path, sensing and other computational geometric approaches [13,19,22,24,26,30]-[48,59,63,64,86,88,92].

Most of the above mentioned methods have three disadvantages. First, the allowed shapes are too restricted to be applicable in general. Secondly, they may fail to find a solution even if there is one. And thirdly, their computational time may be too high.

### 2.3.1 The Potential Field Method

The roadmap and the cell decomposition methods are global methods i.e. these methods search for a free path by first analyzing the connectivity of the whole free space of the robot; and are guaranteed to find a path if it exists. However, their computational time increases exponentially as the degree of freedom of the robot increases [53]. on the other hand, the potential field method which will be the focus of our discussion, is a local approach depending on local information of the resultant force due to an artificial potential induced by the obstacles and goal. The robot is represented as a point mass under the influence of the artificial potential $U$. This potential is usually defined over free-space as the sum of an attractive potential pulling the robot toward the goal configuration, and a repulsive potential pushing the robot away from the obstacles. The robot and the obstacles are assumed to carry positive electic charge while the goal point is assumed to carry a negative charge. The resulting scalar potential field is used to represent the free space.

We have choosen to work on the potential field method because of its speed, though it might fail to find a solution for a small set of hard problems. At the same time, it allows a richer set of objects and motions.

The potential field method is very elegant and can be very efficient requiring no prior model of the obstacles when used in an on-line collision avoidance scheme. Although not as thorough as the graph seaching techniques, the speed of the algorithms and the easy extension to higher dimensions make them excellent alternatives to the graph searching techniques. However, the potential field method has one serious drawback. Since it is essentially a fast optimisation descent scheme, it can get stuck in a local minima other than the goal configuration. For few obstacles

in the configuration space, this problem may not arise. But for a cluttered environment, this problem seriously limits the application of the potential field method.

The local minima are created due to the addition of attractive potential due to the goal and repulsive potentials from several obstacles. Therefore, at certain points in the potential field, the net force on the robot becomes zero, which are local minima of the potential field, and thereby the robot stops at an unintended location.

The potential field approach was pioneered by Khatib [45] who used it in real-time collision avoidance. Khatib first used the FIRAS (force inducing artificial repulsion from a surface) function. Using an analytic description of the obstacle $O : f(q) = 0$ , the repulsive potential is given by

$$U_{rep,O}(q) = \left\{ \begin{array}{ll} \frac{1}{2}\eta\left(\frac{1}{f(q)} - \frac{1}{f(q_0)}\right)^2 & \text{if } f(q) \leq f(q_o) \\ 0 & \text{if } f(q) > f(q_0) \end{array} \right\} \tag{2.17}$$

and the repulsive force is given by

$$\vec{F}_{rep,O} = -\nabla U_{rep,O} = \left\{ \begin{array}{ll} \eta\left(\frac{1}{f(q)} - \frac{1}{f(q_0)}\right)\nabla\left(\frac{1}{f(q)}\right) & \text{if } f(q) \leq f(q_0) \\ 0 & \text{if } f(q) > f(q_0) \end{array} \right\} \tag{2.18}$$

where $q_0$ is a point in the vicinity of the obstacle and $\eta$ is a constant gain. The region of influence of this potential is between $f(q) = 0$ and $f(q) = f(q_0)$. $U_{rep}$ is a non-negative continuous and differentiable function whose value tends to infinity as the robot approaches the obstacle's surface, and becomes negligible beyond that. A plot of this potential and its contour plot is shown in Fig.2.9

Later, Khatib proposed the following potential function using the shortest distance to the

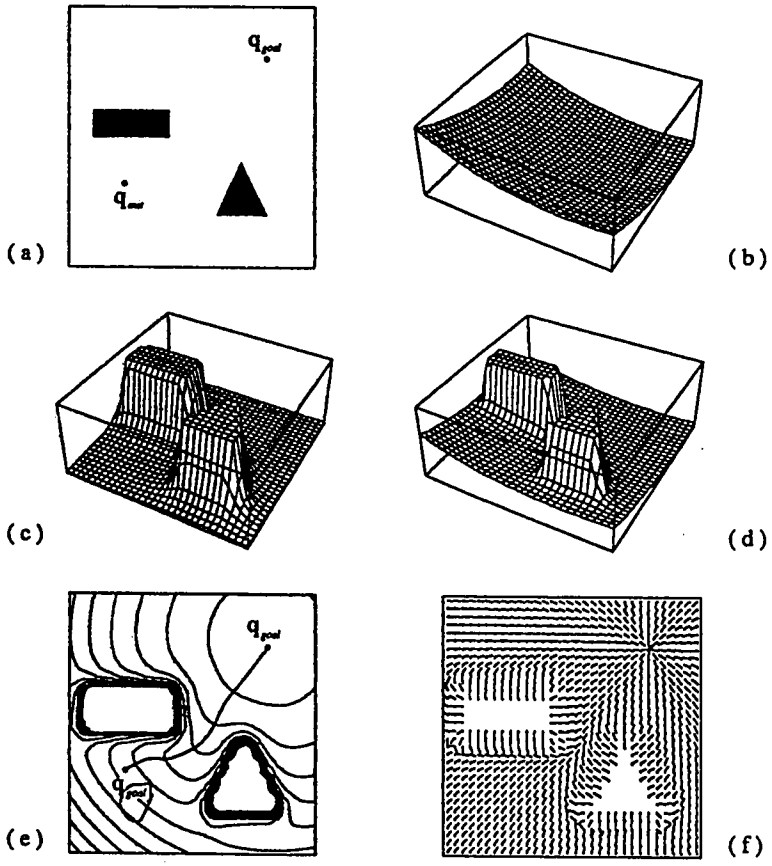


Figure 2.9: The FIRAS potential function: (a) two polygonal obstacles, (b)attractive potential field,(c)repulsive potential field,(d)the sum of the two, (e)contour plot of the total field, (f)vector field of the total potential (adopted from [53]).

obstacle.

$$U_{rep,O}(q) = \left\{ \begin{array}{ll} \frac{1}{2}\eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho(q_0)}\right)^2 & \text{if } \rho(q) \leq \rho_o \\ 0 & \text{if } \rho(q) > \rho_0 \end{array} \right\} \tag{2.19}$$

where $\rho(q)$ denotes the distance from the current position $q$ to the C-obstacle region $CB$ i.e.

$$\rho(q) = min_{q' \in CB} \parallel q - q' \parallel$$

and $\rho_0$ is a positive constant called the distance of influence of the C-obstacle. For a convex CB region, the artificial repulsive force due to $U_{rep,O}$ is given by

$$\vec{F}_{rep,O} = -\nabla U_{rep,O}(q) = \left\{ \begin{array}{ll} \frac{1}{2}\eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q)}\nabla\rho(q) & \text{if } \rho(q) \leq \rho_o \\ 0 & \text{if } \rho(q) > \rho_0 \end{array} \right\} \tag{2.20}$$

The attractive potential due to the goal is a parabolic well and is given by

$$U_{att,goal} = \frac{1}{2}\zeta \parallel q - q_{goal} \parallel = \frac{1}{2}\zeta\rho^2_{goal}(q) \tag{2.21}$$

where

$$\rho_{goal}(q) = \parallel q - q_{goal} \parallel$$

The function $U_{att,goal}$ is everywhere differentiable and attains its minima at the goal where $U_{att,goal}(q_{goal}) = 0$. The attractive force at any configuration $q$ is

$$\vec{F}_{att,goal}(q) = -\nabla U_{att,goal}(q) = -\zeta\rho_{goal}(q)\nabla\rho_{goal}(q) = -\zeta(q - q_{goal}) \tag{2.22}$$

The conic well can also be used as an attractive potential, but it does not posses the stabilizing characteristic of the parabolic well [53].

The total potential on the robot at any position is the sum of the two potentials; subsequently, the total force on the robot is the sum of the corresponding forces i.e.

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{2.23}$$

and

$$\vec{F} = -\nabla U(q) = -\nabla U_{att} - \nabla U_{rep} \tag{2.24}$$

The above two repulsive potential functions proposed by Khatib are not devoid of local minima.

The potential field approach has also appeared in the Russian literature due to the work of Pavlov and Voronin [67]. They proposed the use of the following potential functions:

$$P_1 = Ke^{-\alpha\rho^2}$$

$$P_2 = \frac{K}{1+\alpha\rho^2}$$

$$P_3 = \frac{\alpha}{|\rho|}$$

where $\alpha$, $K$ are positive constants .

$$\rho = \| X - X^* \| \; ; X, X^* \in \Re^n$$

where $X^*$ is the danger point.

To overcome the problem of local minima in the potential field of the robot, Koditchek [71,72] proposed the notion of a *navigation function.* This is a local-minima-free potential function. The goal is the global minimum whose domain of attraction includes the entire subset of $C_{free}$ that is connected to the goal. However, the construction of such a function is difficult except for restricted shapes of obstacles [53]. For the case of a spherical work-space and spherical obstacles defined respectively as

$$W \stackrel{\text{def}}{=} \{q \in \Re^n : \| q \|^2 \leq \rho_0^2\}$$

$$O_j = \{q \in \Re^n : \| q - q_j \|^2 < \rho_j^2\} \; j = 1, 2....N(number\ of\ obstacles)$$

If we define the spherical obstacles functions by

$$\beta_0(q) = - \| q - q_j \|^2 + \rho_0^2$$

$$\beta_j(q) = - \| q - q_j \|^2 + \rho_j^2 \quad for\, j = 1...N$$

then the function

$$\Psi(q) = \frac{\| q - q_{goal} \|^2}{[\| q - q_{goal} \|^k + \beta(q)]^{1/k}} \tag{2.25}$$

where $\beta(q) = \prod_{j=0}^{N} \beta_j$ , is a navigation function provided the constant $k$ exceeds a certain function of the geometrical data.

Because of the difficulty in constructing analytic navigation functions, numerical navigation functions have been proposed. These involve the discretization of the robots free-space into rectanguloid grids and assigning values for the potential function in each grid [53]. Various other methods for tackling the local minima problem have also been proposed with limited success [53].

Volpe and Khosla [47,89] have proposed the use of elliptical and superquadric artificial potentials for obstacle avoidance and approach. These potential functions can closely model arbitrary convex obstacles (eg. rectangular, conic,triangular and etc.), yet they do not generate local minima.

The idea is to generate spherical potentials so as to avoid the creation of local minima when these potentials are added to an attractive well. We will consider the superquadric potentials since they are a generalization of the elliptical potentials. The obstacles are surrounded by a superquadric defined as

$$\left[\left(\frac{x}{f_1(x,y,z)}\right)^{2n} + \left(\frac{y}{f_2(x,y,z)}\right)^{2n}\right]^{\frac{2m}{2n}} + \left(\frac{z}{f_3(x,y,z)}\right)^{2m} = 1 \tag{2.26}$$

where $f_1, f_2, f_3$ are scaling functions and $m, n$ are exponential parameters. If $f_1 = a$, $f_2 = b$, $f_3 = c$, *and* $m = 1$, the superquadric becomes an n-ellipsoid; in two dimensions we have an n-ellipse.

Now, define a contour function by

$$\left(\frac{x}{a}\right)^{2n} + (b/a)\left(\frac{y}{b}\right)^{2n} = 1 \tag{2.27}$$

This function generates spherical contours away from the surface of the object and converges on its surface. Define the Pseudo-distance from the obstacle by

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

where $k$ varies from zero on the n-ellipse to infinity away from it. Therefore, it can serve as a repulsive potential for some $n$ defined by

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

where $\alpha$ is an adjustable parameter. $n$ must vary from infinity to one while $K$ varies from zero to infinity. The repulsive potential is now defined by

$$U_{rep}(x, y) = \eta \frac{e^{-\alpha K}}{K}$$

where $\eta$ is a scaling factor, $\alpha$ determines how fast the potential rises near the object and falls away from it. The potential also has an inverse dependence on the distance.

Warren [90] has used potential functions in a trial and error fashion for global path planning. In his approach, a candidate path is proposed and then modified under the influence of the artificial potential field until a collision-free path is determined. By considering the whole path, the problem of becoming trapped in a local minimum is greatly reduced. The potential field is divided into two parts: the potential inside the C-space obstacle, $U_{in}$ and a lower potential $U_{out}$ outside the C-space obstacle. The two potentials are given by

$$U_{in} = U_{max}\left(1 - \frac{R_{in}}{R_{max}}\right) + U_{offset} \tag{2.28}$$

and

$$U_{out} = \frac{1}{2} U_{offset} \left( \frac{1}{1 + R_{out}} \right) \tag{2.29}$$

Where $U_{max}$ is the maximum potential allowed, $R_{in}$ is the distance from the current position to the centroid of the obstacle, $R_{max}$ is the distance from the centroid to the farthest point on the boundary of the obstacle. $U_{offset}$ is an additional potential applied to all points on the obstacle used to produce more of a penalty for crossing an obstacle. With an additional potential function to penalize the length of each path segment, the total potential is minimized in a discretized C-space to determine the shortest safe path.

Boreinstein [10] has used a combination of certainty grids for obstacle representation, and potential field method for on-line obstacle avoidance. The models of the obstacles are derived from sensory data.

Tilove [85] used the distance to the nearest object as sensed by a ring of range sensors as a repulsive potential. Path planning is then done using two algorithms. A hill climbing algorithm which corresponds to steepest descent in the potential field, and a force control algorithm which utilizes the velocity and acceleration to determine the path.

Huang and Ahuja [38] proposed the use of the following potential function in an algorithm comprising of a global planner and a local one.

$$P_{rep} = \frac{1}{\delta + \sum_{i=1}^{m} [g_i(x) + |g_i(x)|]} \tag{2.30}$$

where $g_i(x) \leq 0$ $i = 1, 2...m$ represent the set of inequalities describing the convex obstacle and $\delta$ is a very small positive penalty parameter. This potential function has the advantage that it can handle convex polytopes, although it is not continuosly differentiable.

Kim and Khosla [50] have proposed the use of hydrodynamic harmonic potentials for real-

time obstacle avoidance using the panel method. The most important property of the harmonic potentials is that they are free from local minima. The total potential due to obstacles, goal, and uniform flow is given by

$$\phi(x,y) = \phi_{uniform} + \phi_{goal} + \sum_{j=1}^{m} \phi_j = -U(x\cos\alpha + y\sin\alpha) + \frac{\lambda_g}{2\pi} + \sum_{j=1}^{m} \frac{\lambda_j}{2\pi} \int_j \log R_j dl_j \quad (2.31)$$

where $\phi_{uniform}$ is a harmonic potential.

$R_g = \sqrt{(x-x_g)^2 + (y-y_g)^2}$ g for goal

$R_j = \sqrt{(x-x_j)^2 + (y-y_j)^2}$ j for panel

$\alpha$ is the angle between the direction of uniform flow and the X-axis

$m$ - the number of panels representing an object.

$\lambda_g$ , $\lambda_j$ are the strengths of the goal and panel respectively.

$U$ - is the strength of uniform flow.

Once the strengths of the panels have been determined [50], the velocities of the point robot are determined from

$$u_x(x,y) = -\phi_x = u\cos\alpha - \frac{\lambda_g}{2\pi}\frac{\partial}{\partial x}\log R_{ij} - \sum_{j=1}^{m} \frac{\lambda_j}{2\pi} \int_j \frac{\partial}{\partial x} \log R_{ij} dl_j \quad (2.32)$$

$$u_y(x,y) = -\phi_y = u\sin\alpha - \frac{\lambda_g}{2\pi}\frac{\partial}{\partial y}\log R_{ij} - \sum_{j=1}^{m} \frac{\lambda_j}{2\pi} \int_j \frac{\partial}{\partial y} \log R_{ij} dl_j \quad (2.33)$$

The disadvantage of this method is the enormous amount of precomputation involved due to the large number of panels required to approximate an obstacle (see Fig.2.10) . Secondly, there may arise the presence of structural local minima.

Hashimoto et al. [36] have proposed the use of electrostatic Laplace potentials for obstacle avoidance using sliding mode control, and finally, in [39,52] potential functions are used for on-line collision avoidance.

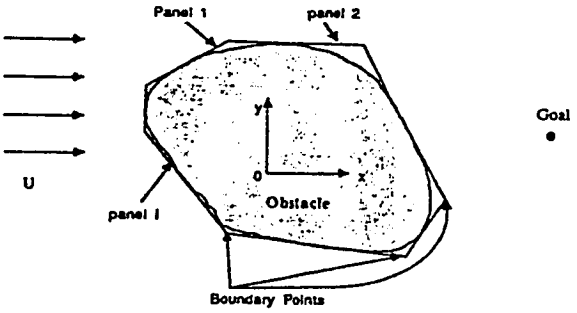


Fig. 5. Panel method.

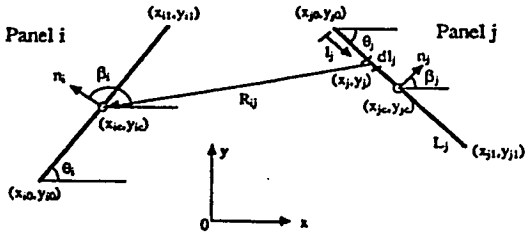


Figure 2.10: Construction of panels from an obstacle (adopted from 50)

# Chapter 3

# THE COLLISION DETECTION ALGORITHMS

## 3.1 Introduction

This chapter deals with the trasformation of objects features as they move in three dimensional space, as related to the collision detection algorithms. We shall see, based on the representations in chapter 2, how an object can be described as it translates and rotates from an initial location to another. These transformations will form an important basis for our algorithms. Once these ideas have been firmly established, the collision detection algorithms will be introduced.

Before we derive the expressions for representing a dynamic object, we will review some of the fundamental transforms in rigid body kinematics.

## 3.2 Review of Transformations:

Consider the reference frames $UVW$ and $XYZ$ shown in Fig.3.1(a). $UVW$ is assumed to be attached to an object, while $XYZ$ is the world coordinate frame. The transformation between the two frames is given by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Where $(x, y, z)$ are the coordinates of any point on the object as measured in the $XYZ$ frame, and $u, v, w$ are the coordinates of the same point as measured in the $UVW$ frame.

Now consider the $UVW$ frame rotated by an angle $\theta$ about the $Z-axis$ as shown in Fig.3.1(b). Then, the trasformation between the two frames is given by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = R_{z,\theta} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.1}$$

Similarly, if the $UVW$ frame is rotated around the $X$ or $Y-axis$, the following rotation matrices result[27]

$$R_{X,\alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix} \tag{3.2}$$

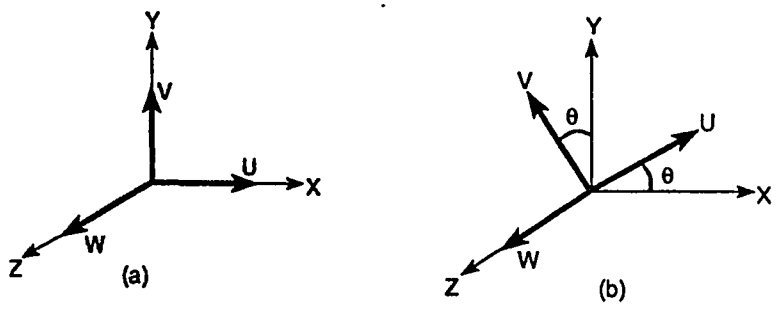


Figure 3.1: Frame rotation

$$R_{Y,\phi} = \begin{bmatrix} cos\phi & 0 & sin\phi \\ 0 & 1 & 0 \\ -sin\phi & 0 & cos\phi \end{bmatrix} \tag{3.3}$$

The matrices $R_{X,\alpha}$, $R_{Y,\phi}$, and $R_{Z,\theta}$ are orthonormal and satisfy

$$R^{-1} = R^T, \ |R| = 1$$

Now consider traslating the origin of the $UVW$ frame to a point $P(p_x, p_y, p_z)$ as shown in Fig.3.2. This transformation can be represented as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{3.4}$$

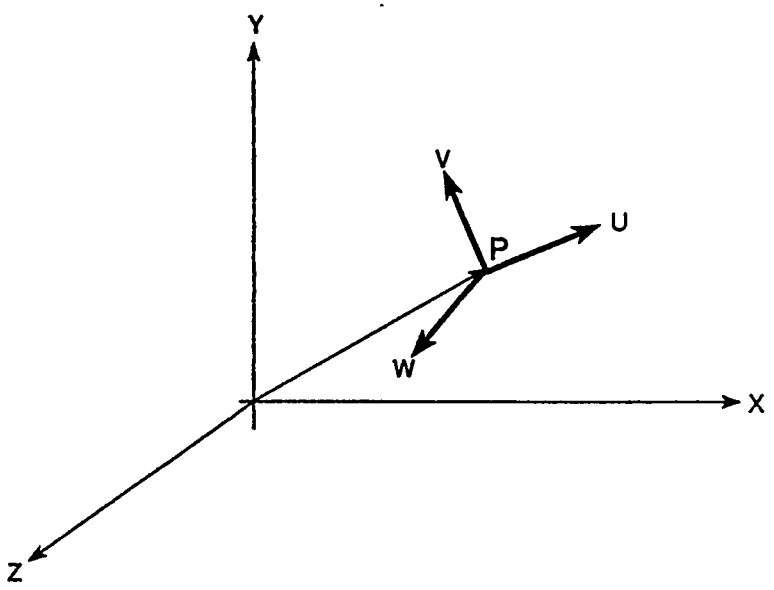


Figure 3.2: Frame translation and rotation

To combine both rotation and traslation of the $UVW$ frame about the $XYZ$ frame we have:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_{XYZ} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{3.5}$$

The above transformation can be represented by a single transform matrix called *homogeneous transform* with dimension $(4 \times 4)$. The coordinates of every point are then represented in homogeneous coordinates with dimension $n + 1$ [21,27,69]

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3\times3} & P_{3\times1} \\ 0 & 1 \times 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \tag{3.6}$$

For example, a rotation of $\theta$ about the $Z$-axis followed by a traslation to $P(p_x, p_y, p_z)$ is

represented by the homogeneous transform :

$$T = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\theta & -S\theta & 0 & p_x \\ S\theta & C\theta & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

where $C\theta = cos\theta$ and $S\theta = sin\theta$.

In general, any complex transformation can be represented as :

$$T = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

where $\mathbf{n} = (n_x, n_y, n_z)$, $\mathbf{s} = (s_x, s_y, s_z)$ and $\mathbf{a} = (a_x, a_y, a_z)$ are the components of the unit vectors of the $UVW$ frame on the $XYZ$ frame respectively [14].

### 3.2.1 Euler Angles Representation of Orientation

One set of Euler angles representation for orientation is the *Roll,Pitch* and *Yaw (RPY)*. It corresponds to the following rotations in sequence:

- A rotation of $\psi$ about the $OX - axis$ ($R_{X,\psi}$)-*Yaw*
- A rotation of $\theta$ about the $OY - axis$ ($R_{Y,\theta}$)-*Pitch*
- A rotation of $\phi$ about the $OZ - axis$ ($R_{Z,\phi}$)-*Roll*

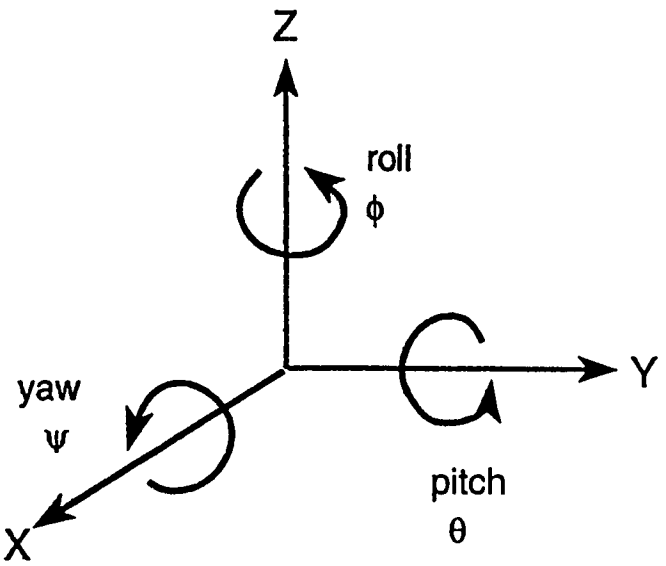


Figure 3.3: Roll, pitch and Yaw Euler angles

The three rotations are shown in Fig.3.3

The composite rotation matrix resulting from these three rotations is given by:

$$R_{\phi,\theta,\psi} = R_{Z,\phi}R_{Y,\theta}R_{X,\psi} = \begin{bmatrix} C\phi C\theta & C\phi S\theta S\psi - S\phi C\psi & C\phi S\theta C\psi + S\phi S\psi \\ S\phi C\theta & S\phi S\theta S\psi + C\phi C\psi & S\phi S\theta C\psi - C\phi S\psi \\ -S\theta & C\theta S\psi & C\theta C\psi \end{bmatrix} \tag{3.9}$$

The position and orientation of the object is represented by the $4\times4$ homogeneous transformation matrix given by:

$$T_{RPY} = \begin{bmatrix} C\phi C\theta & C\phi S\theta S\psi - S\phi C\psi & C\phi S\theta C\psi + S\phi S\psi & p_x \\ S\phi C\theta & S\phi S\theta S\psi + C\phi C\psi & S\phi S\theta C\psi - C\phi S\psi & p_y \\ -S\theta & C\theta S\psi & C\theta C\psi & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

For a rotation about an arbitrary axis $\mathbf{r} = (r_x, r_y, r_z)$ by an angle $\phi$, the orientation matrix is given by :

$$R_{r,\phi} = \begin{bmatrix} r_x^2 V\phi + C\phi & r_x r_y V\phi - r_x r_y V\phi - r_z S\phi & r_x r_z V\phi + r_y S\phi \\ r_x r_y V\phi + r_z S\phi & r_y^2 V\phi + C\phi & r_y r_z V\phi - r_x S\phi \\ r_x r_z V\phi - r_y S\phi & r_y r_z V\phi + r_x S\phi & r_z^2 V\phi + C\phi \end{bmatrix} \tag{3.11}$$

Where $V\phi = 1 - \cos\phi$

## 3.3 Transformation of Object Features

Now consider an object represented by the system

$$\Lambda_i x \leq b_i \tag{3.12}$$

If the object is translated by a vector $P_i = (p_x, p_y, p_z)$ and at the same time rotated around any axes, the new coordinates $x'$ of a point $x$ on the object are given by:

$$x' = R_i x + P_i \tag{3.13}$$

Where $R_i$ is the corresponding rotation matrix.

Substituting equation.(3.13) in equation.(3.12) and rearranging, the new geometry of the object is given by the following system

$$\Lambda_i R_i^{-1}(x' - P_i) \leq b_i \tag{3.14}$$

**Example 1**

Consider the object $O_1 \in \Re^2$ shown in Fig.3.4(a). It is represented by the system:

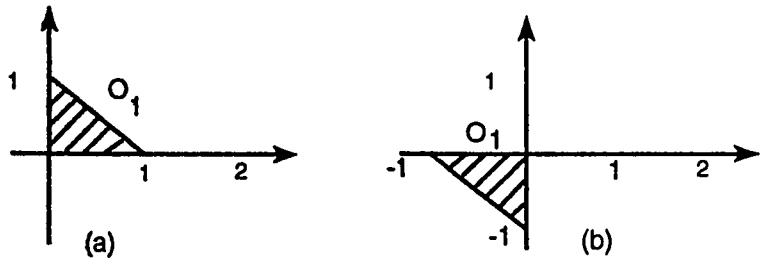


Figure 3.4: Example 1.

$$x + y \leq 1$$

$$x \geq 0$$

$$y \geq 0$$

Rotating it by 180° around the $Z - axis$ results in the geometry shown in Fig.3.4(b). This is uniquely determined by using equations(3.14) and (3.1)

$$\begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

or

$$-x - y \leq 1$$

$$x \leq 0$$

$$y \leq 0$$

Which is the required geometry. The case of translation is simple.

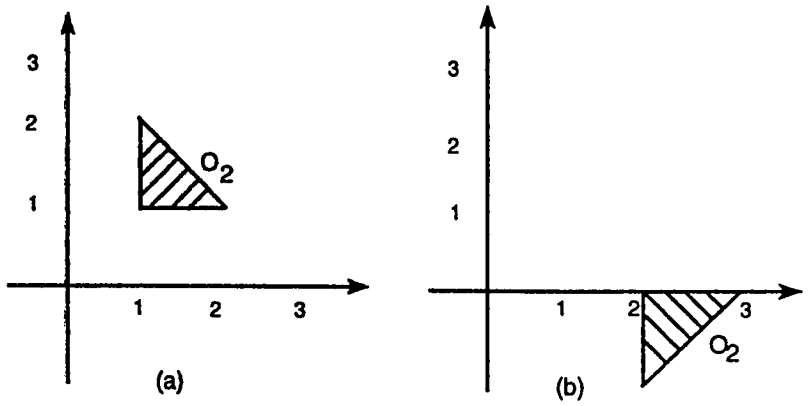


Figure 3.5: Example 2.

For an object represented as a convex hull of vertices, translation and rotation of the object defined by (1.2) results in the new set

$$co\mathbf{X} = \{x = \sum_{j=1}^{l} \lambda_j T_j x_j \ : \sum_{j=1}^{l} \lambda_j = 1 \ ; \lambda_j \geq 0 \ ; j = 1...l\} \tag{3.15}$$

Where the vertices $x_j$ are defined in homogeneous coordinates[14] and $T_j$ is the homogeneous transformation matrix representing the translation and rotation of the object.

**Example 2**

Consider the object $O_2 \in \Re^2$ shown in Fig.3.5(a), rotating it by 180° around the $X - axis$ and translating it by $P = (1\,,\,1)^T$, results in the geometry in Fig.3.5(b)

Initially,

$$O_2 : X = \lambda_1 \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} + \lambda_3 \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

;

$$\lambda_1\ \lambda_2\ \lambda_3 \geq 0.$$

After the rotation and the translation,

$$O_2 : X = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left\{ \lambda_1 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} 1 \\ 2 \\ 0 \\ 1 \end{bmatrix} + \lambda_3 \begin{bmatrix} 2 \\ 1 \\ 0 \\ 1 \end{bmatrix} \right\}$$

or

$$X = \lambda_1 \begin{bmatrix} 2 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} 2 \\ -1 \\ 0 \\ 1 \end{bmatrix} + \lambda_3 \begin{bmatrix} 3 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1;$$

$$\lambda_1\ , \lambda_2\ , \lambda_3 \geq 0.$$

## 3.4 The Algorithms

To develop an algorithm for collision detection, let us represent the path of the objects by their translation vectors $P(t)$, $P_i(t), i = 1,2...N$ and their orientations by the orientation matrices $R(t)$, $R_i(t), i = 1,2..N$. These two sets of functions define the location functions of the objects. Furthermore, since the objects are in motion, each element of these functions is a function of time.

To determine collision between the objects using the first representation, we represent $A$ at any point in time and space by

$$\Lambda R^{-1}(t)(x - P(t)) \leq b$$

and the obstacles by

$$\Lambda_i R_i^{-1}(t)(x - P_i(t)) \leq b_i \ , i = 1, ...N$$

Solving the above two systems of inequalities together for $i = 1, ...N$ at discrete points in time, will give the collision points if there are any.

# Algorithm I

*Initialization step*

1) Input $t_s, t_f \ , \Delta t, \Lambda, \ b, \ \Lambda_i, \ b_i \ ; i = 1, ....N$

2) Define $P(t)$, $R(t)$, $P_i(t)$, $R_i(t)$, $i = 1, ..N$

3) set $i = 1, \ t = t_s$ and go to the main step

*Main steps*

4) Determine $P(t), \ R(t), \ P_i(t), R_i(t)$.

5) Solve the system

$$\begin{bmatrix} \Lambda R^{-1}(t) \\ \Lambda_i R_i^{-1}(t) \end{bmatrix} x \leq \begin{bmatrix} b + \Lambda R^{-1}(t) P(t) \\ b_i + \Lambda_i R^{-1}(t) P_i(t) \end{bmatrix}$$

for a feasible point using a standard linear programming algorithm with zero objective function.

6) Report $x(t)$, the collision point, if the system has a solution. Otherwise, the objects dont collide at this instance.

7) If $i = N$ go to (8), otherwise, increment $i$ by 1 and goto (4).

8) If $t = t_f$ stop, otherwise, increment $t$ by $\Delta t$, set $i = 1$ and go to (4).

End

To develop the second algorithm using the second representation of the objects, note that, two objects will interfere at any instance if there is a point that belongs to the convex hulls of both objects.

i.e. if

$$A(t) : co\mathbf{X} = \{x = \sum_{j=1}^{n_A} \lambda_j T(t) x_j \ : \ \sum_{j=1}^{n_A} \lambda_j = 1 \ ; \lambda_j \geq 0 \, \forall j\}$$

and

$$B_i(t) : co\mathbf{Y} = \{y = \sum_{j=1}^{n_i} \gamma_j T_j(t) y_j \ : \ \sum_{j=1}^{n_i} \gamma_j = 1 \ ; \gamma_j \geq 0 \, \forall j\}$$

Then

$$A(t) \cap B_i(t) \neq \emptyset \ iff \ \exists x \in A(t) \ and \ y \in B_i(t) \mid x = y.$$

# Algorithm II

## *Initialization step*

1) Input the vertices of the objects as columns of the matrices $O$, $O_1, ....O_N$.

2) Input $t_s, t_f, \Delta t$

3) Define the location matrices of the objects $T(t), T_1(t), T_2(t) .....T_N(t)$

4) set $i = 1$ , $t = t_s$ and go to the main step

*Main step*

5) Determine $T(t), T_i(t)$ and update the vertices of the objects $O(t) = T(t) \times O,\ O_i(t) = T_i(t) \times O_i$

6) Solve the system

$$\bar{O}(t)\lambda - \bar{O}_i(t)\gamma = 0$$

$$E^T \lambda = 1 \quad \lambda_j \geq 0, \forall j$$

$$E^T \gamma = 1 \quad \gamma_j \geq 0, \forall j$$

for a feasible point using a standard linear programming algorithm with zero objective function. Where $E = (1, 1, ....1)^T$ and $\lambda = (\lambda_1, ....\lambda_{n_A})^T$, $\gamma = (\gamma_1, \gamma_2 .... \gamma_{n_i})^T$, $\bar{O}(t)$ *and* $\bar{O}_i(t)$ are the first three rows of $O(t)$ *and* $O_i(t)$ respectively.

7)If the system has a solution, Compute the collision point from $x_c = O(t) \times \lambda$. Otherwise, there is no collision with the object at this instance.

8) If $i = N$ goto 9, otherwise, increment $i$ by 1 and goto (5)

9) If $t = t_f$ stop, otherwise, increment $t$ by $\Delta t$ set $i = 1$ and go to (5)

End

# Chapter 4

# COMPUTATIONAL EXPERIENCE ON THE ALGORITHMS

## 4.1 Introduction

In this chapter, we discuss implementation and computational experience on the proposed collision detection algorithms. The algorithms are also compared in terms of simplicity of implementation, computational speed/complexity, and accuracy. In this regard, problems in two and three dimensions were solved. We begin the chapter with a brief review of linear programming techniques which form the backbone of our collision detection algorithms.

### 4.1.1 Linear Programming

The general linear program is an optimization problem that minimizes (maximizes) a linear objective function subject to linear inequality and/or equality constraints. Mathematically rep-

resented as

$$Min.\ Cx$$

$$s.t.\ \ Dx = d$$

$$x_l \leq x \leq x_u$$

where $x \in \Re^n$, $C\,is\ (1 \times n)$, $D\,is\,(m \times n)$ and $d\,is\ (m \times 1)$

The three main approaches for solving the above problem are:

1. The simplex method[7]

2. The ellipsoidal methods[3,7]

3. The interior point methods[3,7]

Theoretically, the ellipsoidal and the interior point methods are the only polynomial ones. However, in practice, the average case performance is much more important than the worst case behaviour. And for the problem of finding or identifying a feasible solution to a system of linear inequalities which we are interested in, the simplex method has been well tested and has proven to have good performance.

## 4.2 Implementation

In our implementation of the collision detection algorithms, the simplex method was used to solve the systems in steps 5 and 6 of the two algorithms respectively. A standard simplex-method-based subroutine for linear programming is provided by the 'MATLAB optimization toolbox' [94], and this was used in the implementation of our algorithms. Furthermore, the algorithms require as

inputs the constraints matrices representing the objects for Algorithm I, and the vertices of the objects for Algorithm II, the initial time, the final time, the time increments and the motions of the objects. The algorithms have been coded in MATLAB language, and the above parameters are read as inputs to the programs. The path is inputed in terms of the coefficients of the path components $p_x$, $p_y$, $p_z$. parameterized in time. A listing of the codes is provided in Appendices $A, B, C, D$.

## 4.3 Two dimensional Example

The two dimensional example that was considered is shown in Fig.4.1. The case of the object $O$ moving only was considered. The constraint matrices of the objects were determined from the interceps of the edges of the objects projected on to the X and Y axes. The following are the constraint matrices of the objects 1-10 including the moving object O.

$$O: \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} X \leq \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

$$O_1: \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 2 & 0 \end{bmatrix} X \leq \begin{bmatrix} 6 \\ -4 \\ 0 \\ 8 \end{bmatrix}$$

$$O_2: \begin{bmatrix} 0.6 & -1 & 0 \\ -1 & 1 & 0 \\ -2 & -1 & 0 \\ 1 & 1 & 0 \end{bmatrix} X \leq \begin{bmatrix} -1.8 \\ 3 \\ -8 \\ 7.5 \end{bmatrix}$$

$$O_3: \begin{bmatrix} 1 & -2 & 0 \\ -1 & -2 & 0 \\ -2 & -1 & 0 \\ -1 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} X \leq \begin{bmatrix} 3 \\ -12.5 \\ -14.5 \\ 3 \\ 12 \end{bmatrix}$$

$$O_4: \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 0 \end{bmatrix} X \leq \begin{bmatrix} 13 \\ -3 \\ 4.2 \\ -14 \end{bmatrix}$$

$$O_5: \begin{bmatrix} 1 & 1.1 & 0 \\ 1 & -3.5 & 0 \\ -1.2 & -1 & 0 \\ -1 & 2.2 & 0 \end{bmatrix} X \leq \begin{bmatrix} 17.6 \\ -7 \\ -16 \\ 6 \end{bmatrix}$$

$$O_6: \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ -1.1 & -1 & 0 \\ 2 & -2 & 0 \\ 1 & 1 & 0 \\ -1 & 1 & 0 \end{bmatrix} X \leq \begin{bmatrix} 8 \\ -6.8 \\ -14.5 \\ 1 \\ 16 \\ 2 \end{bmatrix}$$

$$O_7: \begin{bmatrix} -1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix} X \leq \begin{bmatrix} -3.7 \\ 5.6 \\ -6 \\ 8 \end{bmatrix}$$

$$O_8: \begin{bmatrix} 0.7 & -1 & 0 \\ 1 & 1.1 & 0 \\ 0 & 1 & 0 \\ -1.8 & -1 & 0 \end{bmatrix} X \leq \begin{bmatrix} -5 \\ 11 \\ 8 \\ -10 \end{bmatrix}$$

$$O_9: \begin{bmatrix} 1 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} X \leq \begin{bmatrix} -6.7 \\ -2.5 \\ 11 \end{bmatrix}$$

$$O_{10}: \begin{bmatrix} -2 & 1 & 0 \\ 1 & 1.05 & 0 \\ -1 & 0 & 0 \end{bmatrix} X \leq \begin{bmatrix} 1 \\ 15.7 \\ -9.2 \end{bmatrix}$$

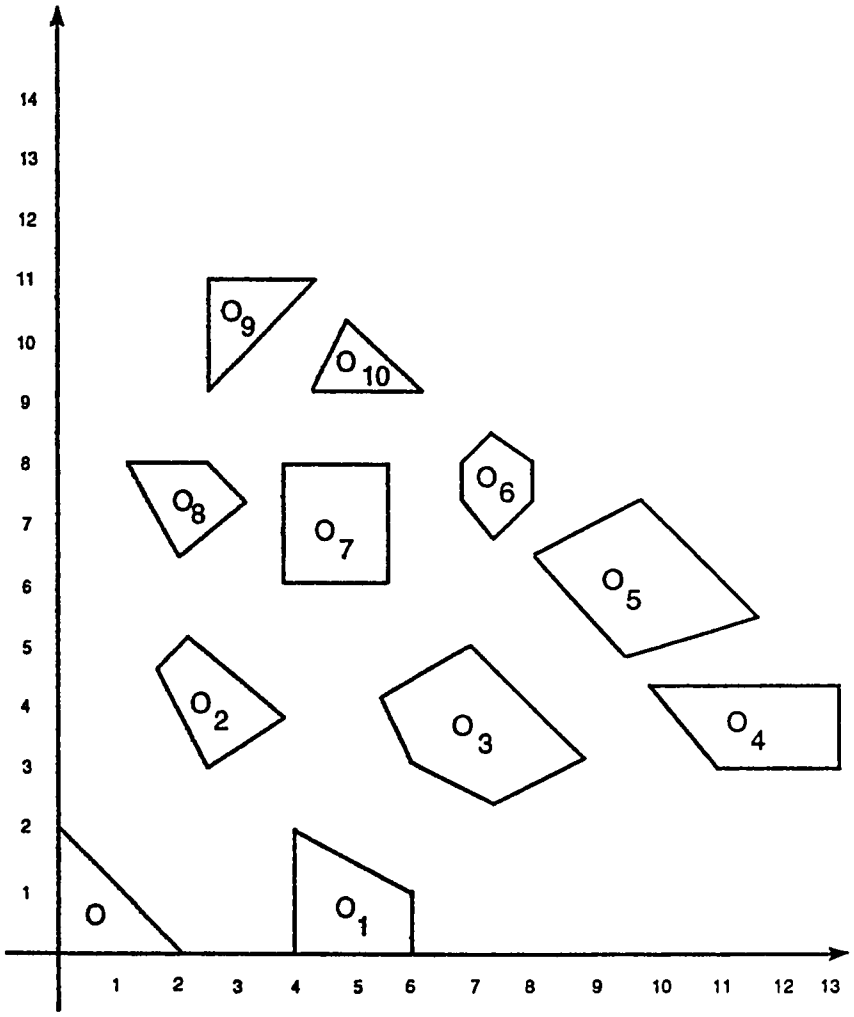


Figure 4.1: A two dimensional example

These were inputed into Algorithm I (Appendix A and C). Similarly, the vertices of the objects were inputed as columns of the following matrices in homogeneous coordinates into Algorithm II (Appendices B ,D)

$$O: \begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$O_1: \begin{bmatrix} 4 & 6 & 6 & 4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$O_2: \begin{bmatrix} 2.5 & 3.7 & 2.2 & 1.7 \\ 3 & 3.7 & 5.2 & 4.7 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$O_3: \begin{bmatrix} 7.4 & 6 & 5.5 & 7 & 8.8 \\ 2.5 & 3.1 & 4.2 & 5 & 3.2 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$O_4: \begin{bmatrix} 11 & 13 & 13 & 9.8 \\ 3 & 3 & 4.3 & 4.3 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$O_5: \begin{bmatrix} 9.5 & 11.7 & 9.7 & 8 \\ 4.8 & 5.5 & 7.3 & 6.5 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$O_6: \begin{bmatrix} 7.3 & 8 & 8 & 7.4 & 6.8 & 6.8 \\ 6.8 & 7.3 & 8 & 8.5 & 8 & 7.3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$O_7: \begin{bmatrix} 5.6 & 5.6 & 3.8 & 3.8 \\ 6 & 8 & 8 & 6 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$O_8: \begin{bmatrix} 2 & 3.2 & 2.5 & 1.21 \\ 6.5 & 7.3 & 8 & 8 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$O_9: \begin{bmatrix} 2.5 & 2.5 & 4.3 \\ 9.2 & 11 & 11 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$O_{10}: \begin{bmatrix} 4.3 & 4.9 & 6.2 \\ 9.2 & 10.4 & 9.2 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Two paths were considered for this example. One linear, and the other parabolic. Also, in the first set of experiments, only pure translational motions were considered, and in the second set of experiments, simultaneous translations and rotations were considered along the two paths.

$$path_1 : p_x = 5t, \quad p_y = 5t$$

$$path_2 : p_y = 5t + 5t^2, \quad p_y = 5t + 5t^2$$

The initial reference point on the object O was the origin (0,0) and the final position was the point (10,10). A time step of 0.1 units was used for the speed of 5 units/s. For higher speed of the moving object, a criterion for choosing the time step $\Delta t$ is to make sure that the time step is less than the time taken by the moving object (at its maximum speed) to pass the minimum dimension of all the other objects. This can be expressed by

$$\Delta t < \frac{minimum\ object\ dimension}{maximum\ speed}$$

The results of the simulations are tabulated on tables 4.1-4.8. Furthermore, the average computational time against the complexity of the obstacles are plotted in Figs.4.3-4.6.

A three dimensional problem involving two objects(polytopes) $I$ which is diamond shaped and static, and $J$ which is a unit cubic, was also solved. The vertices of the two objects with

respect to their centers are respectively

$$I: \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$J: \begin{bmatrix} 0.5 & -0.5 & 0.5 & 0.5 & 0.5 & -0.5 & -0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & 0.5 & -0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & 0.5 & -0.5 & -0.5 & 0.5 & -0.5 & -0.5 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The path on which $J$ was considered to move was elliptical and is given by:

$$p_x = 3\cos\pi t; \quad p_y = \sin\pi t; \quad p_z = 0 \ \ \forall\, 0 \leq t \leq 1$$

This problem was given to our Algorithm II and the time taken on the IBM/486-DX machine was 2.86 seconds.

Further, to study the accuracy of the algorithms, the scenario in Fig.4.2 was constructed. Algorithm I was used for this test. The objects here are thinner, and the effect of changing the time interval from 0.1 to 0.8 was studied. The results are tabulated on Table 4.9. Again, only object $O$ was considered moving on a linear path at 45° to the horizontal axis. It can be noticed from the table that as the time step is increased, the algorithm takes less time to compute all possible collisions. However, this is achieved at the expense of the reliability of the algorithm. It can further be observed that, up to 0.5 time steps, the algorithm detected all the collisions with objects $O_1$, $O_2$, $O_3$,and $O_5$, except at 0.3 where the object $O$ just misses the obstacle $O_1$. This is due to the pencil nature of the edge of the obstacle. But beyond 0.5, the algorithm misses

more frequently. The result of this experiment applies automatically to Algorithm II with only a speed difference.

## 4.4 Conclusion

It has been found out that Algorithm I is more efficient than Algorithm II in terms of lower computational time. However, Algorithm II requires less preparatory effort since it requires as inputs the vertices of the objects. Algorithm I represents the objects as convex polyhedra defined by systems of linear inequalities while Algorithm II represents the objects as convex polyhedra defined as convex hulls of a finite number of vertices. The two algorithms were developed to be used whenever any of the representations is available. Although the first representation is more difficult to derive, Algorithm I is faster to use.

Both algorithms can treat the general case of a three dimensional object translating and rotating on an arbitrary path (not necessarily linear). From the limited experiments, both algorithms have shown average linear computational complexities. Finally, it has been observed that increasing the time interval for testing the collisons has the effect of speeding-up the performance of the algorithms at the expense of decreasing their reliability of detecting all possible collisions.

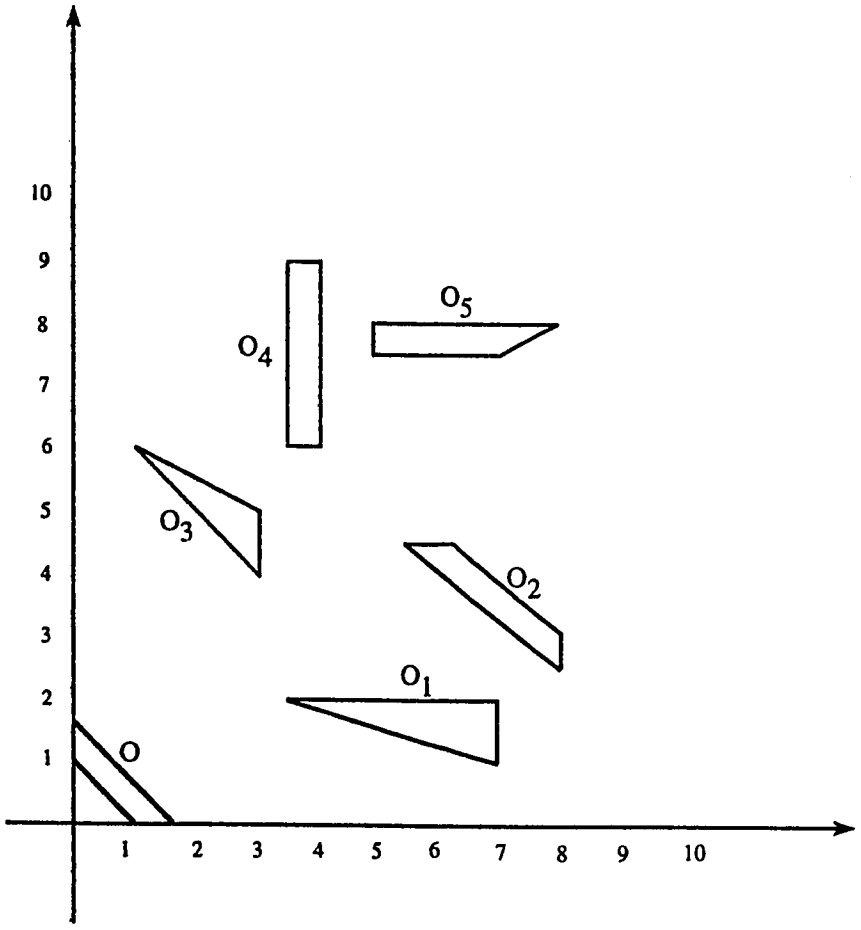


Figure 4.2: An example to study the effect of changing the time interval

Table 4.1: Algorithm I Linear path with translation

| *No. of Objects* | *No. of edges* | *Time(s)* |
|---|---|---|
| 1 | 4 | 4.23 |
| 2 | 8 | 8.13 |
| 3 | 13 | 12.58 |
| 4 | 17 | 16.10 |
| 5 | 21 | 19.94 |
| 6 | 27 | 23.78 |
| 7 | 31 | 27.29 |
| 8 | 35 | 31.47 |
| 9 | 38 | 35.31 |
| 10 | 41 | 40.43 |

Table 4.2: Algorithm I Linear path with translation and rotation

| *No. of Objects* | *No. of edges* | *Time(s)* |
|---|---|---|
| 1 | 4 | 6.97 |
| 2 | 8 | 10.77 |
| 3 | 13 | 15.11 |
| 4 | 17 | 18.51 |
| 5 | 21 | 22.36 |
| 6 | 27 | 26.20 |
| 7 | 31 | 29.55 |
| 8 | 35 | 33.40 |
| 9 | 38 | 36.80 |
| 10 | 41 | 41.52 |

Table 4.3: Algorithm I Parabolic path with translation

| *No. of Objects* | *No. of edges* | *Time(s)* |
|---|---|---|
| 1 | 4 | 2.25 |
| 2 | 8 | 4.29 |
| 3 | 13 | 6.48 |
| 4 | 17 | 8.18 |
| 5 | 21 | 10.11 |
| 6 | 27 | 12.20 |
| 7 | 31 | 13.84 |
| 8 | 35 | 15.82 |
| 9 | 38 | 17.74 |
| 10 | 41 | 19.45 |

Table 4.4: Algorithm I Parabolic path with translation and rotation around X-axis

| *No. of Objects* | *No. of edges* | *Time(s)* |
|---|---|---|
| 1 | 4 | 2.31 |
| 2 | 8 | 4.23 |
| 3 | 13 | 6.48 |
| 4 | 17 | 8.30 |
| 5 | 21 | 10.22 |
| 6 | 27 | 12.14 |
| 7 | 31 | 13.89 |
| 8 | 35 | 15.82 |
| 9 | 38 | 17.52 |
| 10 | 41 | 19.39 |

Table 4.5: Algorithm II Linear path with translation

| *No. of Objects* | *No. of edges* | *Time(s)* |
|---|---|---|
| 1 | 4 | 6.43 |
| 2 | 8 | 12.41 |
| 3 | 13 | 19.44 |
| 4 | 17 | 25.21 |
| 5 | 21 | 31.37 |
| 6 | 27 | 39.00 |
| 7 | 31 | 44.49 |
| 8 | 35 | 50.36 |
| 9 | 38 | 55.47 |
| 10 | 41 | 62.28 |

Table 4.6: Algorithm II Linear path with translation and rotation

| *No. of Objects* | *No. of edges* | *Time(s)* |
|---|---|---|
| 1 | 4 | 6.43 |
| 2 | 8 | 12.31 |
| 3 | 13 | 19.39 |
| 4 | 17 | 24.99 |
| 5 | 21 | 30.98 |
| 6 | 27 | 38.45 |
| 7 | 31 | 44.16 |
| 8 | 35 | 50.26 |
| 9 | 38 | 55.09 |
| 10 | 41 | 59.54 |

Table 4.7: Algorithm II Parabolic path with translation

| *No. of Objects* | *No. of edges* | *Time(s)* |
|---|---|---|
| 1 | 4 | 3.35 |
| 2 | 8 | 6.43 |
| 3 | 13 | 10.11 |
| 4 | 17 | 13.12 |
| 5 | 21 | 16.09 |
| 6 | 27 | 19.93 |
| 7 | 31 | 23.13 |
| 8 | 35 | 25.93 |
| 9 | 38 | 28.57 |
| 10 | 41 | 31.37 |

Table 4.8: Algorithm II Parabolic path with translation and rotation around X-axis

| *No. of Objects* | *No. of edges* | *Time(s)* |
|---|---|---|
| 1 | 4 | 3.29 |
| 2 | 8 | 6.48 |
| 3 | 13 | 9.94 |
| 4 | 17 | 12.85 |
| 5 | 21 | 15.98 |
| 6 | 27 | 19.83 |
| 7 | 31 | 23.23 |
| 8 | 35 | 26.04 |
| 9 | 38 | 28.62 |
| 10 | 41 | 31.14 |

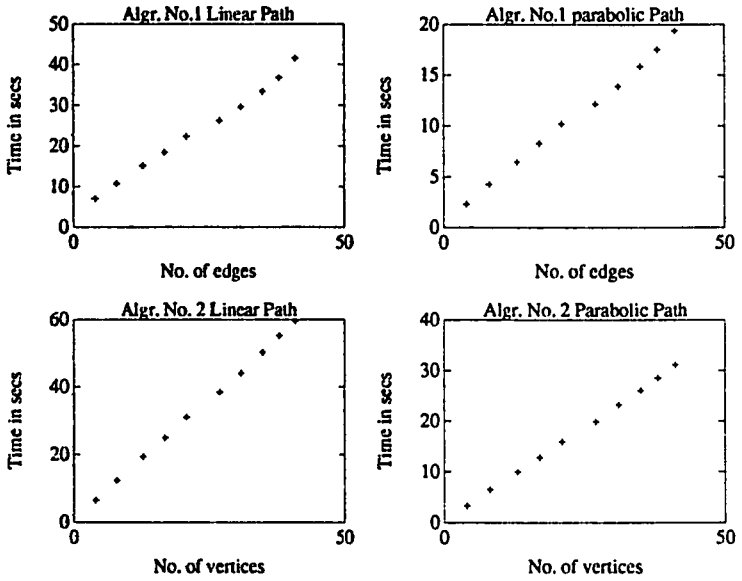


Figure 4.3: The case of translation only

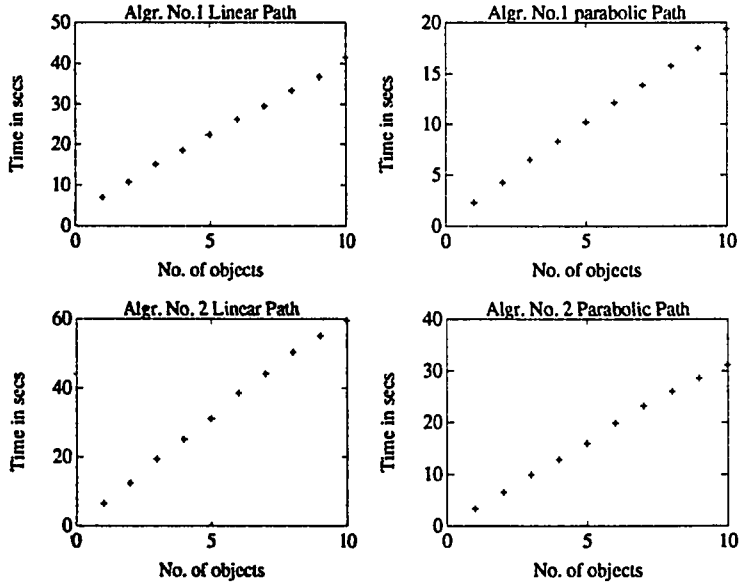


Figure 4.4: The case of translation only

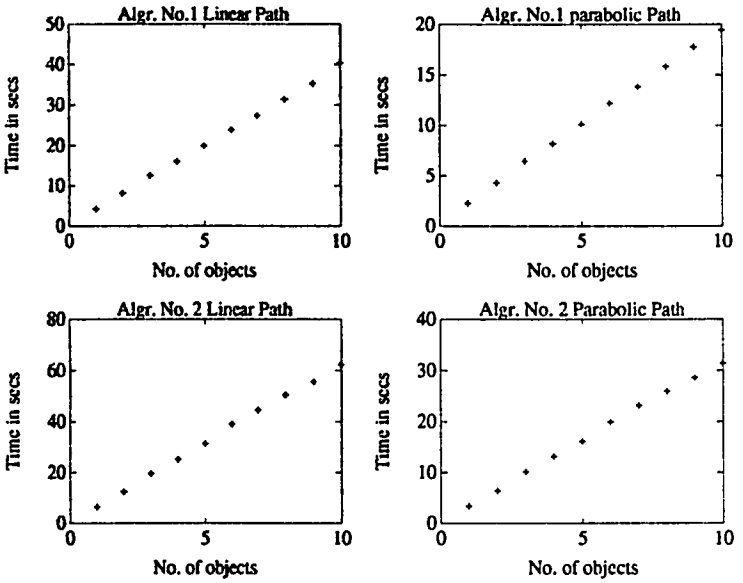


Figure 4.5: The case of simultaneous translation and rotation

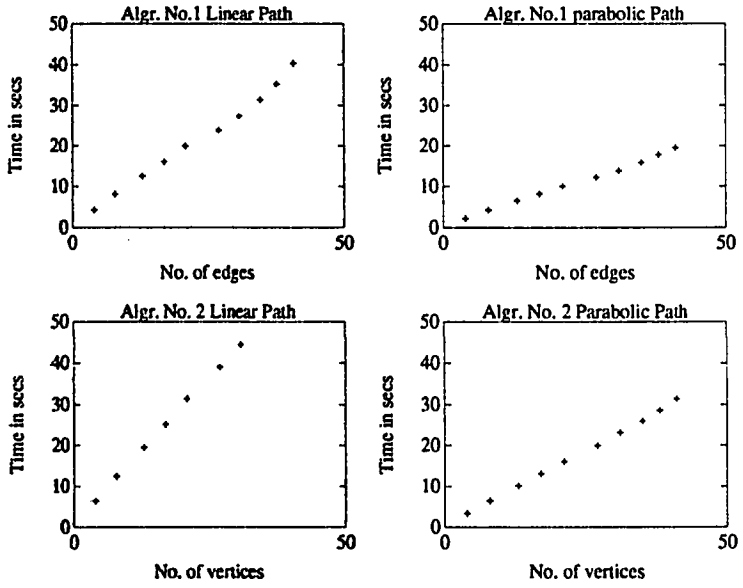


Figure 4.6: The case of simultaneous translation and rotation

Table 4.9: Effect of changing time interval $\Delta t$

| $\Delta t$ | $O_1$ | $O_2$ | $O_3$ | $O_4$ | $O_5$ | *Time(s)* |
|---|---|---|---|---|---|---|
| 0.1 | X | X | X | * | X | 146.81 |
| 0.2 | X | X | X | * | X | 67.33 |
| 0.3 | – | X | X | * | X | 45.43 |
| 0.4 | X | X | X | * | X | 35.04 |
| 0.5 | X | X | X | * | X | 26.97 |
| 0.6 | – | – | X | * | X | 21.92 |
| 0.7 | – | – | X | * | X | 19.28 |
| 0.8 | – | – | – | * | X | 16.59 |

Where X - Algorithm detects a collision

– Algorithm misses a collision

$*$ - No possible collision with object.

# Chapter 5

# APPLICATION TO A ROBOTIC PROBLEM

## 5.1 Introduction

In this chapter, the collision detection algorithms are used in a robotic application, a two link manipulator moving among stationary obstacles, to ascertain their practicality. However, recall that the algorithms are valid for motions (paths) specified in operational (cartesian) space while manipulator trajectories are normally specified in joint space. The transformation of obstacle geometries and their motions from task space to joint space is a very difficult task and is not one-to-one. But, robot direct kinematics can be used to transform joint trajectories to operational space. Therefore, we will review robot kinematics and trajectory planning in the beginning of the chapter, and then apply the collision detection algorithms to a two link manipulator example.

## 5.2 Robot Kinematics

Kinematics is the branch of mechanics that deals with motion (position, velocity, acceleration and higher derivatives) without regard to the forces causing them. In robot kinematics, we are interested in two problems: direct and inverse kinematics problems which are defined below respectively:

### Direct kinematics

For a given manipulator, given the joint angle vector $q(t) = (q_1(t), q_2(t), .....q_n(t))^T$ and the geometrical link parameters, where $n$ is the number of degrees of freedom, what is the position and orientation of the end-effector of the manipulator with respect to a reference coordinate system ?

### Inverse Kinematics

Given the desired position and orientation of the end-effector of the manipulator and the geometric link parameters with respect to a reference coordinate system, what are the various joint angles required to attain this configuration ?

### Trajectory Planning

Trajectory planning is the generation of the time history of the various joint positions, velocity and acceleration that take the robot from an initial position to a final position along a determined or prescribed path.

Trajectories can also be planned in cartesian space. However, we will be mainly concerned here with joint trajectories. This involves, interpolating or approximating the desired path by a class of polynomial functions and generating a sequence of time-based "control set points" for the control of the manipulator. There are various schemes to manipulator joint trajectory planning: 4-3-4, 3-5-3 (polynomial) and 5-cubic-spline interpolated trajectories [21,27,69].

To connect the above problems with collision detection, we now solve an example involving a two link manipulator moving among few obstacles. To do this, we will assume that we are given an initial representation of the manipulator links in the form of polyhedral sets as discussed in chapter 1 and on which the collision detection algorithms are based. We will also assume that we are given the joint trajectories and we have to transform them into equivalent cartesian trajectories using the robot forward (direct) kinematics. This will enable us to find a sequence of new manipulator links representations along the trajectories. Then we apply our collision detection schemes to determine any possible collisions with the obstacles.

### 5.2.1 Kinematic Equations of the two link Manipulator

Using the Denavit-Hartenberg(D-H) representation of links and joints parameters, and link coordinate system assigment[21,27], the D-H matrix for adjacent coordinate frames, $i$ and $i-1$ for a revolute joint is given by

$$A_i^{i-1} = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

where $\alpha_i$, $a_i$, $d_i$ are constant link parameters and $\theta_i$ is variable. For the two link manipulator example shown in Fig.5.1, $\alpha_1 = \alpha_2 = 0$ , $a_1 = a_2 = 0.5m$ then the homogeneous coordinate transformation matrices are given by

$$A_1^0 = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

$$A_1^2 = \begin{bmatrix} C_2 & -S_2 & 0 & l \\ S_2 & C_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

$$A_2^0 = A_1^0 A_2^1 = \begin{bmatrix} C_{12} & -S_{12} & 0 & lC_1 \\ S_{12} & C_{12} & 0 & lS_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

where $C_i = cos\theta_i$, $S_i = sin\theta_i$, $C_{ij} = cos(\theta_i + \theta_j)$ and $S_{ij} = sin(\theta_i + \theta_j)$ .The matrices $A_1^0$ and $A_2^1$ transform any point on link $i$ whose coordinates are defined with respect to the reference frame of link $i$, to the reference frame of link $i-1$.

Now assume that the manipulator is initially in the horizontal position with both links stretched as shown in Fig.5.2. We can approximate the links by rectangles defined by the fol-

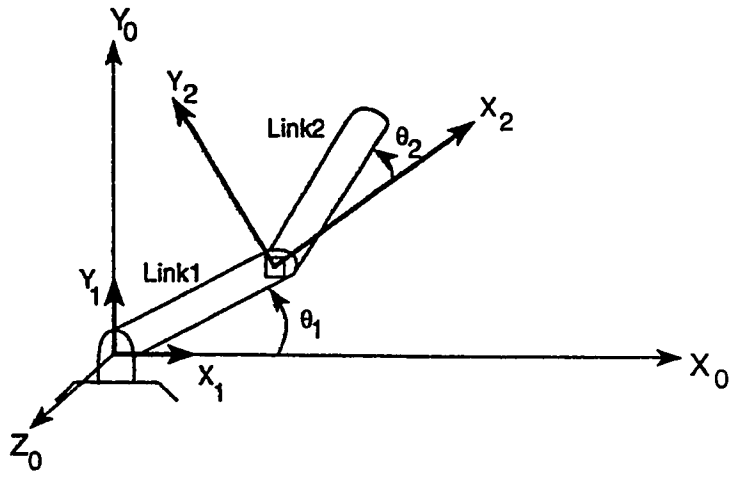


Figure 5.1: A two link manipulator

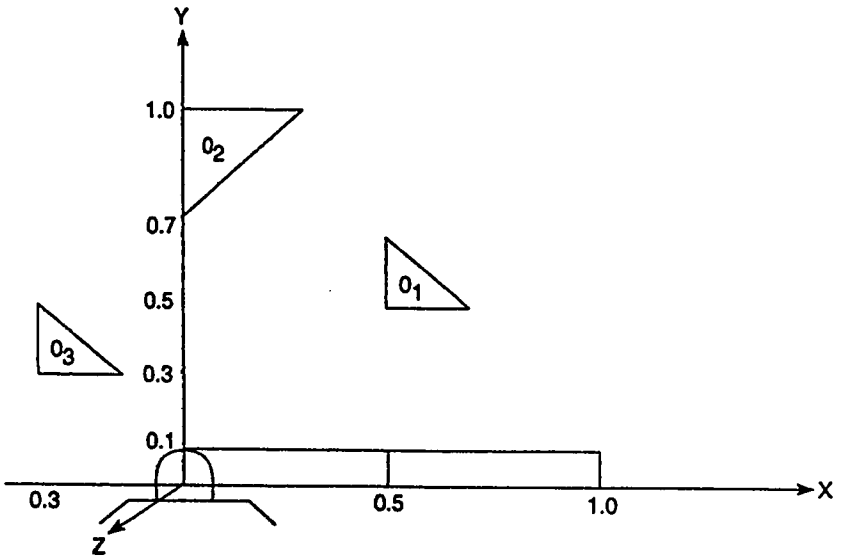


Figure 5.2: A two link manipulator example

lowing systems of inequalities.

$$link_1 : \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0.5 \\ 0 \\ 0.1 \end{bmatrix} \tag{5.5}$$

$$link_2 : \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} -0.5 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{5.6}$$

The obstacles $O_1$, $O_2$, and $O_3$ are similarly represented by the following systems:

$$O_1 : \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1/12 & 1/13 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} -0.5 \\ -0.5 \\ 1 \end{bmatrix} \tag{5.7}$$

$$O_2 : \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 0 \\ 1 \\ -0.7 \end{bmatrix} \tag{5.8}$$

$$O_3 : \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 0.3 \\ -0.3 \\ 0.2 \end{bmatrix} \tag{5.9}$$

Table 5.1: Manipulator Collision Detection

| | *Link 1* | *Link 2* |
|---|---|---|
| $O_1$ | X | t=1.5 (0.701,0.5) |
| $O_2$ | X | t=5.0 (0,5.0) |
| $O_3$ | t=10 (-0.1,3.0) | t=6.2 (-2.228,0.425) |

## 5.3 Simulation

The example of simple manipulator trajectories given by:

$$\theta_1 = \frac{\pi}{20}t, \quad \theta_2 = \frac{\pi}{10}t$$

will be considered.

A time step of 0.1 was used and Algorithm I was applied. The results of the simulation are presented on Table 5.1 below: Where X - No collision

The above table indicates the time and location of the earliest collision with each object.

## 5.4 Conclusion

The collision detection algorithm I, developed has been applied to a practical problem of a plane two-link manipulator moving amongst planar convex obstacles. The algorithm has detected accurately the collision of link-2 with objects $O_2$,$O_2$ and $O_3$, and the collision of link-1 with object $O_3$ only, while moving on the above simple circular trajectories (Table 5.1). This can be verified from Fig.5.2.

# Chapter 6

# PATH PLANNING

## 6.1 Introduction

A number of industrial applications require finding of a minimum length path in the presence of polyhedral obstacles from a starting point to a destination point. Path length minimization of an arbitrary route is helpful in improving the productivity of a process and reducing costs. Typical applications include path planning for manipulator and mobile robots, routing of oil, gas and water piping systems, communication cables etc.

The shortest path between two given points in three dimensions consists of a set of straight lines with their vertices located on the obstacles (see Fig.2.7). The global problem consists of finding the sequence of edges through which the shortest path passes. Searching for the correct sequence of edges is computationally expensive due to the combinatorial nature of the search process. Moreover, in robotic application, to servo a robot on such a path will require the velocity of the robot to instantaneously go to zero at the vertices. This presents a serious problem in motion control.

In this chapter, we will consider the path planning problem for the case of a point mass moving in a space where there is accurate information about the nature, position and orientation of the static configuration-space obstacles (CS-obstacles). Furthermore, we shall assume that the obstacles are not interpenetrating, i.e. they are separated by some distance, or are atmost only touching. We shall introduce a new approach to the potential field method which is essentially a penalty function method but has the remarkable feature that there is no lacal minima in the potential field. The only global minimum is the goal configuration whose domain of attraction is over the whole of $C_{free}$. We shall also propose a new approach based on expanding spheres for optimizing the path in a depth-first fashion, that can be used with any potential or penalty function.

We begin by introducing the requirements desired for a potential function, and then discuss some of the draw-backs of ealier approaches before we introduce our own approach.

## 6.2 Requirements for an Artificial Potential Function

In this section we look at the properties required for a continuous function to serve as potential function. These requirements have motivated the development of the new potential function, as most of the potential functions proposed so far, have fallen short of these requirements.

Khosla and Volpe [47] have summarized the properties of a repulsive artificial potential function as follows :

1. It should have spherical symmetry for large distances from the obstacle so that no local minima is created when this potential is added to other potentials (e.g. an attractive well).

2. It should mimick the obstacle surface at close distance so as to maximize the robots free-

space.

3. It's range of influence should be limited to the vicinity of the obstacle so as not to affect the robots motion away from the obstacle.

4. It should be a continuosly diffentiable function of class $C^n[0,\infty)$ where $n \geq 2$.

An attractive potential is usually a quadratic well which has good stabilizing properties and gives a constant gain when used in a feed-back control. A conic well also serves as a good attractive potential but does not have the stabilizing properties of the quadratic well.

## 6.3 Problems with Earlier Approaches

From the literature review presented in Chapter 1, the FIRAS potential used by Khatib[45] does not satisfy properties (1) and (4), and hence is not free from the problem of local minima when added to an attractive well or other potentials.

The superquadric artificial potentials proposed by Khosla and Volpe [47,89] are very attractive and satisfy almost all the requirements of the ideal. However, the inverse dependence of the potential with the pseudo-distance '$k$' from the obstacle retains a finite value for the potential at distances away from the obstacles. Moreover, the choice of the scaling parameters and the decay constant for the potential still remains a trial-and-error problem.

Again, the harmonic potentials proposed by Kim and Khosla [50] eliminate the local minima problem but do not optimize the path. Secondly, as the complexity of the obstacles increases, the number of panels required to represent them increases and the computational cost of the algorithm may become very high.

The potential function proposed by Huang and Ahuja [38] has the advantage that it can handle convex polyhedral obstacles but it still remains finite at distances away from the obstacle. Therefore, the generation of local minima in the potential field cannot be ruled out. Although, the use of a global planner and a local planner has given rise to a powerful algorithm.

The notion of a navigation function proposed by Khoditchek and Rimon [71,72], is achievable for only limited classes of objects shapes and configuration-space.

## 6.4 The New Potential Function

Consider an analytic description of a smooth convex object (such as a sphere or ellipsoid) given by

$$g(x) \leq 0 \; ; \; g(x) \in C^n[0,\infty), \;\; x \in \Re^n \; ; n = 2,3 \tag{6.1}$$

Then the function

$$p(x) = \mu max(0, -g(x))^r \tag{6.2}$$

where $\mu$ is a very large penalty parameter and $r \geq 2$, is everywhere zero outside the object, and attains its maximum in the interior of $g(x) \leq 0$. Hence, such a function can serve as a very good potential function. Furthermore, the addition of such potentials arising from so many objects in a configuration-space, will not engender the creation of any local minimum.

For a point robot moving amongst $N$ stationary obstacles defined by $g_i(x) \leq 0 \; ; \; i = 1,2,...N$, a useful navigation function can be defined by

$$U(x) = \frac{1}{2} \parallel x - x_g \parallel^2 + \mu \sum_{i=1}^{N} max(0, -g_i)^r \tag{6.3}$$

Where $\eta$ is a constant.

The above total potential function is continuous, differentiable and has zero contribution from the second term at locations away from the obstacles. Its remarkable property is described in the following proposition.

**Proposition :** The potential function $U(x)$ given in equation(6.3) has a unique global minimum at the goal configuration whose domain of attraction extends all over $C_{free}$, for all $r \geq 2$.

*Proof:*

The gradient vector of the potential function is given by

$$\nabla U = (x - x_g) + (-1)^r r\mu \sum_{i=1}^{N} max(0, -g_i)^{r-1} \nabla g_i \tag{6.4}$$

Because of the large penalty parameter $\mu$, the interior of the obstacles is a forbidden region for the robot, and hence if we exclude all points in the interior of the obstacles, the second term in the gradient is always zero, and the solution of

$$\nabla U = 0 \; \textit{is uniquely solved by} \; x = x_g$$

The hessian is positive definite at this point, and hence the result •

The only draw-back to the above potential function is that it allows the robot to approach too close to the surfaces of the obstacles. However, this is not a very serious problem as observed in the visibility graph method [56]. The obstacles could be expanded by a small safety factor $\epsilon$ to avoid this problem. Furthermore, this seeming drawback can be used to advantage since certain robot tasks such as docking, parts mating, and more general motion tasks, all require navigation at or along the boundary of the configuration space. Fig.6.1 shows the distribution of the potential due to two circular objects in $\Re^2$ and Fig.6.2 shows the distribution of the potential due to two rectangular objects obtained by approximation with an n-ellipse.

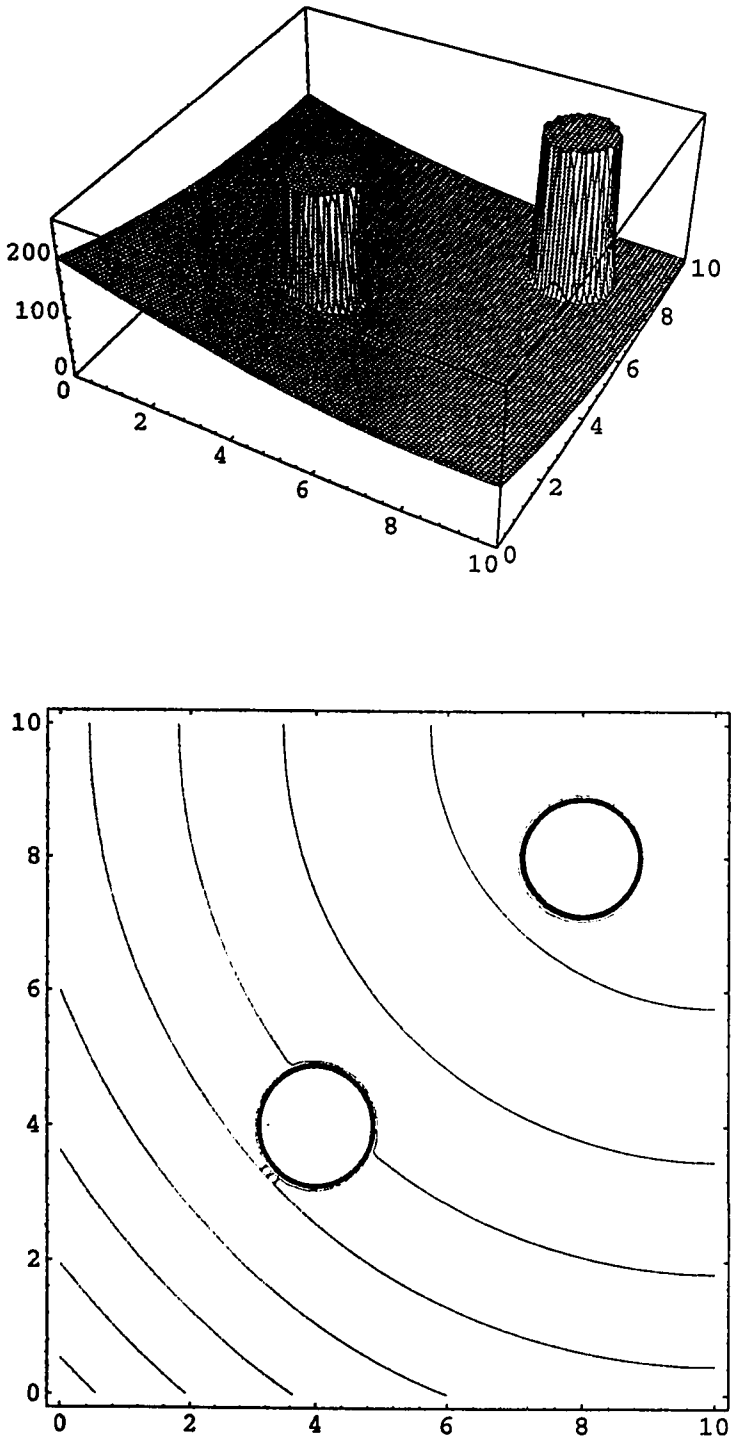


Figure 6.1: (a)Potential field of two circular objects (b)Contour plot of the field

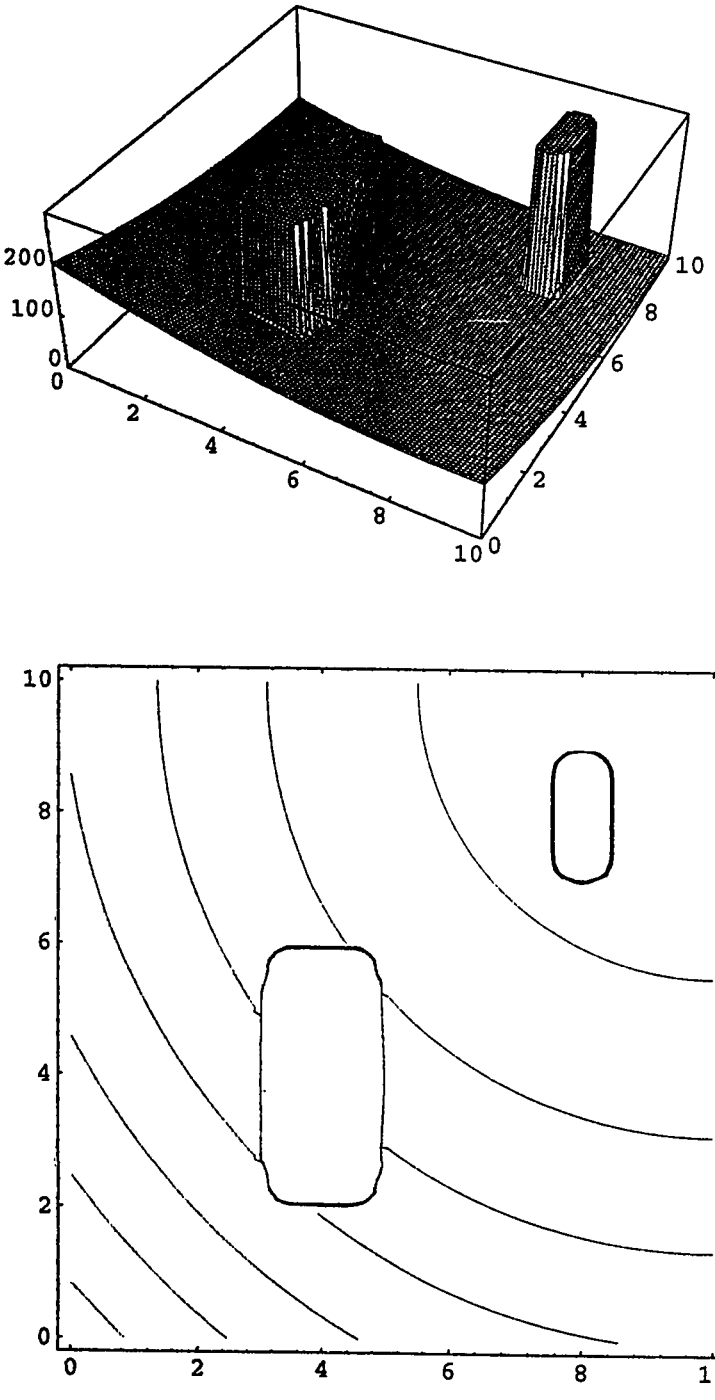


Figure 6.2: (a)Potential field of two rectangular objects (b)Contour plot of the field

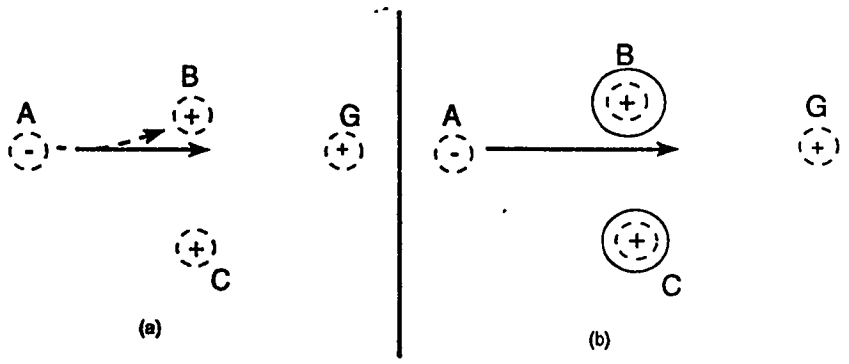


Figure 6.3: A system of electrostatic charges

An analogy of the above potential field to the electrostatic potential can be drawn from Fig.6.3. In the figure, (a) shows a system of three positive charge regions ($B, C$ *and* $G$) and a region of negetive charge ($A$) at the end. It is desired to accelerate the center of the negative charge $A$ to $G$ through the field. To avoid the use of excessively high potential difference between $A$ and $G$, the problem could be solved by screening the regions $B$ and $C$ with metallic surfaces as shown in (b) so that their contribution to the total potential in the region is reduced to zero outside the metallic screens. This is exactly the purpose served by the function $p(x)$.

## 6.5 A New Approach to Path Optimization

Once the total potential function from the goal and obstacles has been constructed, path planning is usually implemented in a depth-first fashion. This involves generating successive path segments starting from the initial configuration along the direction of the negated gradient of the total potential function $U$ at each successive point until the goal is reached. The amplitude of each segment is chosen so that the segment lies in $C_{free}$. Hence the cordinates of the configuration

$x_{i+1}$ attained at the *ith* iteration is given by

$$x_{i+1} = x_i + \delta_i(-\nabla U(x_i)) \tag{6.5}$$

Where $\delta_i$ denotes the length of the *ith* segment and $\vec{\nabla} U$ is the gradient vector of $U$. Usually, the choice of $\delta_i$ is not optimal.

The above method of depth-first planning is a gradient-based approach which uses only first order information about the function. This makes it very inefficient. To solve problem (6.3), Newton's or quasi-Newton methods may be used. However, we are proposing a new scheme that will avail the path planner the utilization of very efficient optimization algorithms in existence. This can be implemented using any optimization package.

Consider the problem

$$Min\ U(x) = \frac{1}{2} \parallel x - x_g \parallel^2 + \mu \sum_{i=1}^{N} max(0, -g_i)^r$$

s.t.

$$\parallel x - x_0 \parallel \leq R(t) \tag{6.6}$$

The additional constraint is an expanding sphere, where $x_0$ is the initial configuration, and $R(t)$ is a time function which represents the radius of the bounding sphere. By the above constraint, problem 6.3 is transformed into a homotopy in R(t), where R(t) goes from zero to $d_g$ or the Euclidean distance from the initial point to the goal point.

The optimality conditions for problem (6.6) are

$$(x - x_g) + (-1)^r r\mu \sum_{i=1}^{N} max(0, -g_i)^{r-1} \nabla g_i - \alpha(x - x_0) = 0$$

$$(x - x_0)^2 = R^2(t)$$

$$\alpha \geq 0 \tag{6.7}$$

The conditions (6.7) are the Karush-Kuhn-Tucker conditions[6] for problem (6.6) and if we ignore the condition $\alpha \geq 0$, then (6.7) is a system of $n+1$ equations in $n+1$ unknowns parameterized by $R(t) \in [0, d_g]$, or a homotopy in $R(t)$.

Newton's method can be used to solve the above system for various values of $R(t) \in [0, d_g]$. Clearly, all the conditions of the implicit function theorem hold, or the solution will constitute a path that is continuous and smooth in $x$ which is the desired objective.

The above has been the theoretical framework for the path planning problem. Implementation can be done either along the same theoretical approach outlined above, where $\alpha \geq 0$ can be taken care of implicitly (i.e. through starting with $\alpha > 0$ and applying the minimum ratio test to make sure it is always nonnegative) or by solving problem 6.6 directly using any of the efficient optimization routines which are readily available nowadays.

Fig.6.4 shows a geometrical interpretation of the above formulation in two dimensions. In each iteration of the path search, the radius of the sphere $R(t)$ is incremented, and the minimum of the potential function is sought within the new sphere(circle). Consequently, a new configuration is generated nearer to the goal and away from the obstacles. The rate of change of $R(t)$ can be typically made equal to the maximum speed of the robot. By a judicious choice of $\dot{R}(t)$, a smooth path can be generated from the initial configuration to the goal configuration without collision with any of the obstacles.

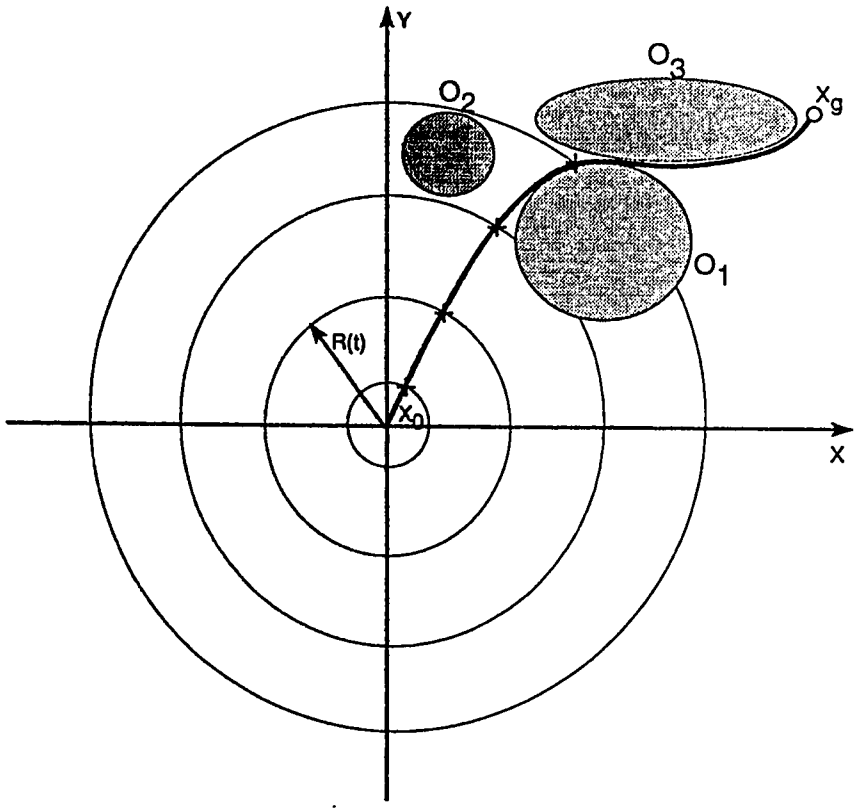


Figure 6.4: Depth-first planning using expanding circles

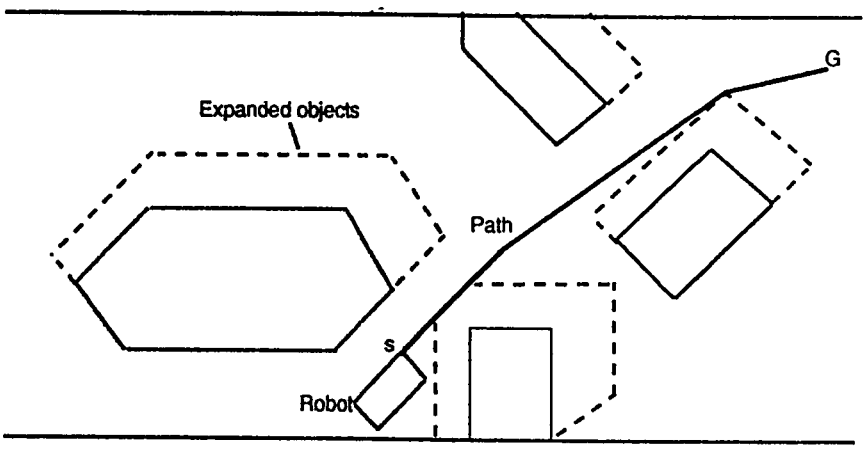


Figure 6.5: Vertex graph path of a polygonal robot through a set of expanded objects

## 6.6 Practical Considerations

In practice, the robot links will have nonzero length, width and thickness; hence, the assumption of a point robot will not work in practice. However, a non-point robot can be transformed into a point robot by expanding the obstacles by the largest radius of the robot. This operation is known as *growing the obstacles by the robot*, and transforms the workspace obstacles into CS-obstacles. This can be achieved using an object growth algorithm [41,53,55]. Although the operation is not exact, it provides a very good practical solution to the above problem. Fig.6.5 shows such an operation for the case of a polygonal robot moving amidst polygonal obstacles.

In Fig.6.6 we show the case of a circular robot amongst circular obstacles. But note that, the growth operation has led to the intersection of the grown obstacles . Depending on the location of the goal configuration, if the robot has to pass in-between the obstacles in order to reach the goal, the robot will get stucked. Yet, the potential field approach has no way of detecting such a problem a priori. A reasonable solution to the above problem is to define a new obstacle which contains the union of the intersecting obstacles. We shall call this new object the

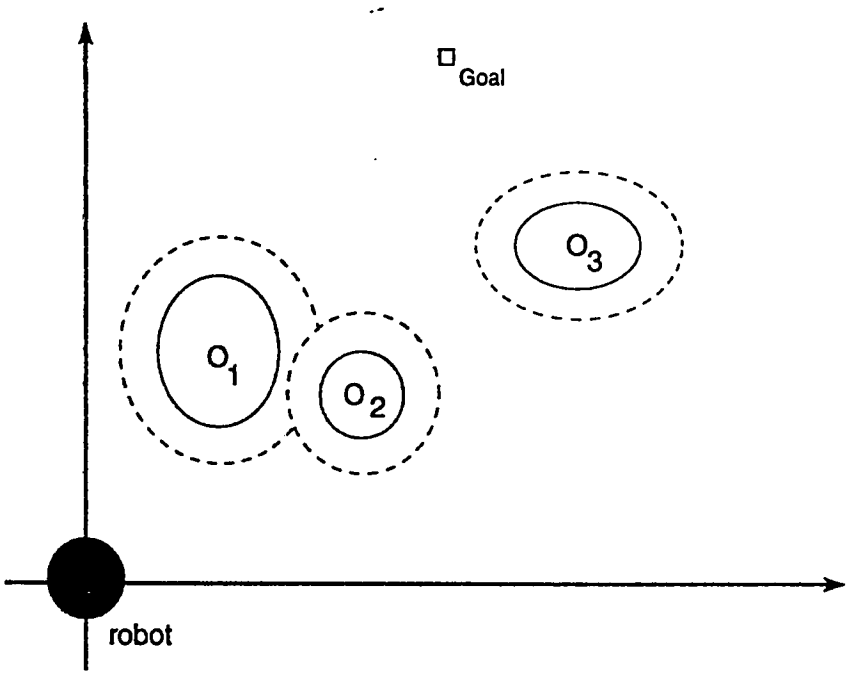


Figure 6.6: A circular robot moving among circular obstacles

*unionizing object.* For the case of the circular obstacles, this new object can be approximated by an n-ellipse defined as follows: let the radii of the grown obstacles be $r_1$, $r_2$ and their centers be $(a_1, b_1)$, $(a_2, b_2)$ respectively. Then the length of the major and minor axes of the unionizing n-ellipsoid $l_1$, $l_2$ respectively, are given by

$$l_1 = (a+b) + \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2}.$$

$$l_2 = max(r_1\,,\, r_2)$$

The center of the n-ellipse $(a, b)$ is similarly given by

$$(a,b) = \frac{1}{2}\left((a_1 - a_2)\,,(b_1 - b_2)\right)$$

Hence the n-ellipse can be defined by

$$\left(\frac{(\bar{x}-a)}{l_1}\right)^{2n} + (\frac{b}{a})^2 \left(\frac{(\bar{y}-b)}{l_2}\right)^{2n} = 1$$

Where $n \geq 1$ is an appropriate shaping parameter.

$$\bar{x} = x\cos\theta - y\sin\theta \ , \ \bar{y} = x\sin\theta + y\cos\theta$$

and

$$\theta = \cos^{-1}\frac{a_1 - a_2}{\sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2}}$$

## 6.7 Simulations

In the simulations, we considered problems in two and three dimensions with circular(spherical) and elliptical obstacles. All the simulations were implemented using the MATLAB Optimization Toolbox routine "CONSTR"[94].

The figures 6.7 -6.12 show the path of a point mobile robot starting from the reference position (0, 0), the origin (without any loss of generality), to the the point (10, 10). Path1 was obtained with three circular obstacles of radii '1' centered at (2,2),(5,5),(8,8) respectively. Path2 was obtained also with circular obstacles of radii '2' centered at (3,3) and (8,8) respectively. Path3 was obtained with two circular obstacles with radii '1' centered at (2,0) and (2,2). The goal in this case was (10,2). Path4 was obtained with a rectangular obstacle approximated by an 4-ellipse with major and minor axes lengths '2' ,'1' respectively, and centered at (5,0). The goal was in this case (10,2). Path5 was obtained using two ellipses centered at (3,3) and (6,6) with length of major axis(along X-axis) and minor axis(along Y-axis) '4', '2' respectively. Path6 was also obtained using ellipses centered at (3,3) and (6,6) but with major axis length '4' along the Y-axis and minor axis length '2' along the x-axis.

Clearly, paths 4 and 5 indicate the failure of the algorithm to handle elliptical obstacles. However, this does not mean that the new potential function proposed has failed to avoid the

obstacles, but it is rather the formulation of the path planning problem in equation (6.6) that has failed. Nevertheless, we still have a remedy to this problem! We propose the replacement of the expanding sphere(circle) constraint in (6.6) with a very narrow expanding ellipsoid(ellipse). We have implemented this back-up procedure for all the above problems, and the results are shown in Figs.6.13-6.18. It has in fact turned out that this procedure yields more optimal paths and solves richer amount of problems. Finally, path13 represent a three dimensional example obtained with a sphere of radius 2 units centered at (5,5,5). The time taken to solve each of the above problems was less than 4.0 minutes on an IBM 486/433 DX machine.

## 6.8 Conclusion

In this chapter, we have presented a new potential function for robot path planning and obstacle avoidance. This potential function has the remarkable feature that, no matter the number of obstacles in the configuration space, no local minima is generated in the potential field. All that is required to generate the potential field using the new potential function, is an analytic description of the obstacles. This is readily available for smooth objects such as spheres and ellipsoids. Nonetheless, convex polytope objects can still be approximated using superquadrics[47].

Furthermore, we have proposed a new approach to path optimization by parameterizing the path as a function of the radius of an expanding sphere. This approach allows the path planner the utilization of efficient optimization routines in existence.

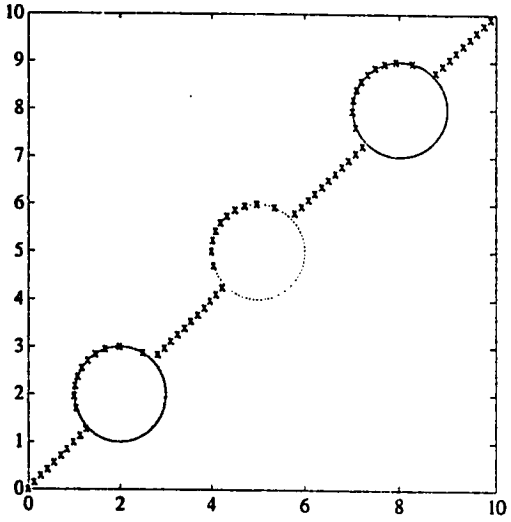


Figure 6.7: Path1

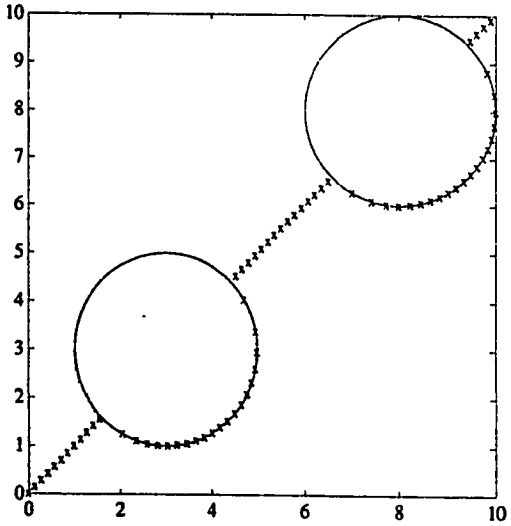


Figure 6.8: Path2

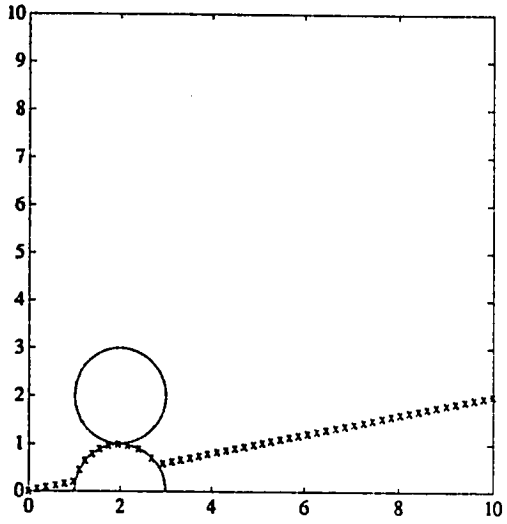


Figure 6.9: Path3

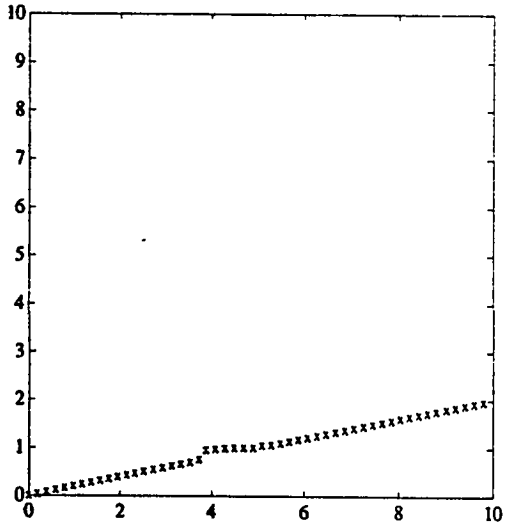


Figure 6.10: Path4

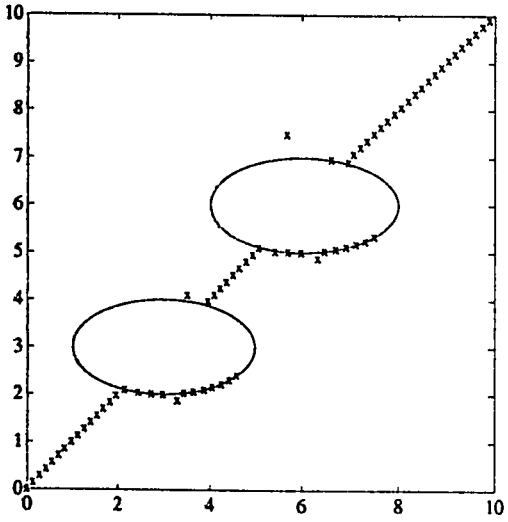


Figure 6.11: Path5

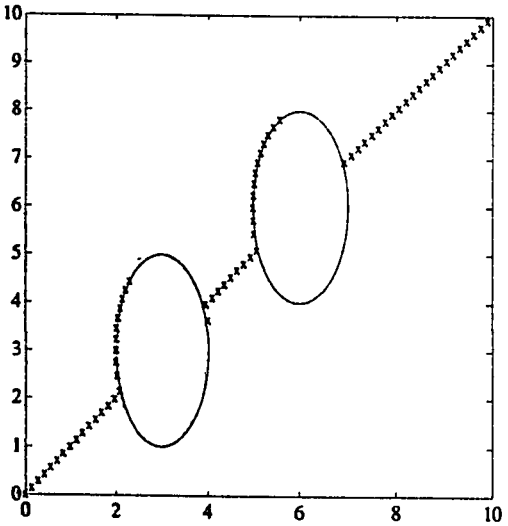


Figure 6.12: Path6

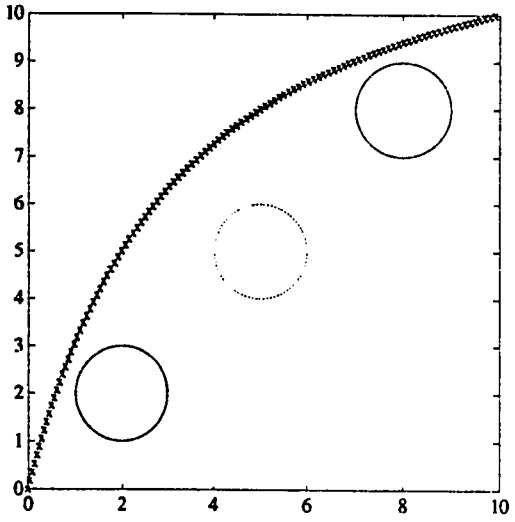


Figure 6.13: Path7

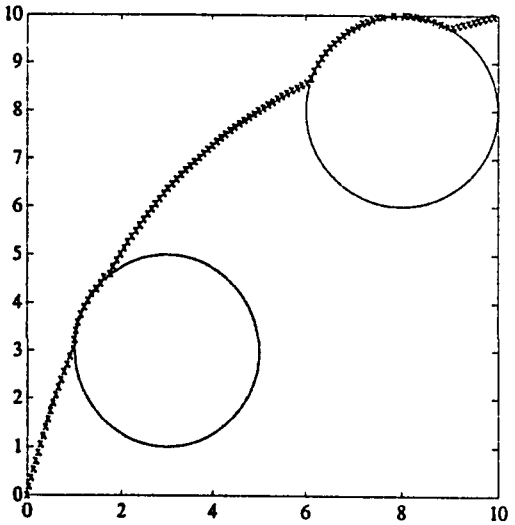


Figure 6.14: Path8

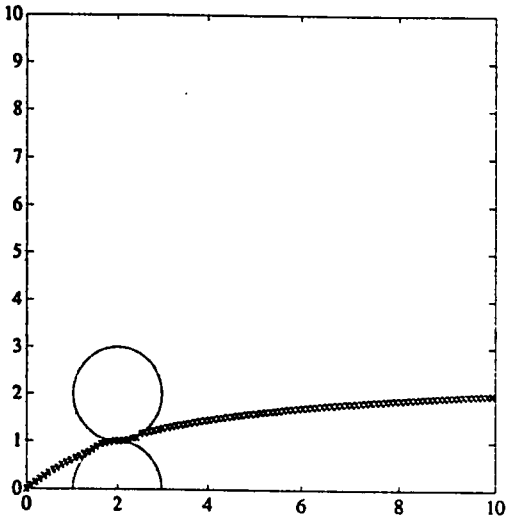


Figure 6.15: Path9

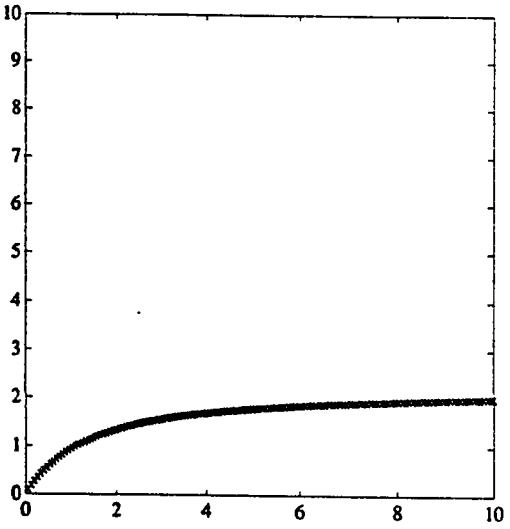


Figure 6.16: Path10

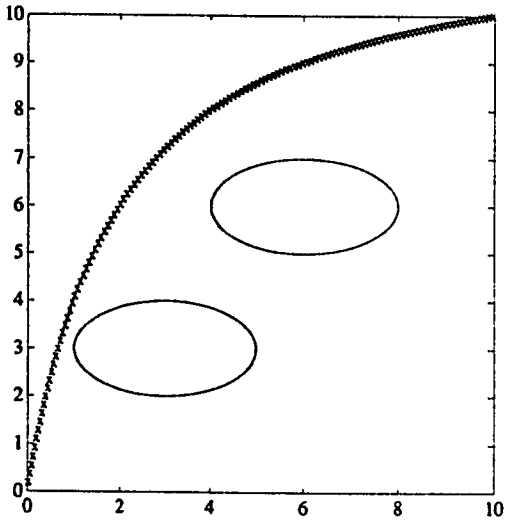


Figure 6.17: Path11

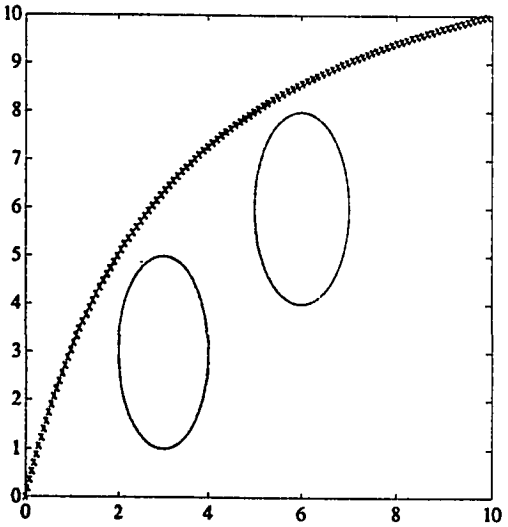


Figure 6.18: Path12

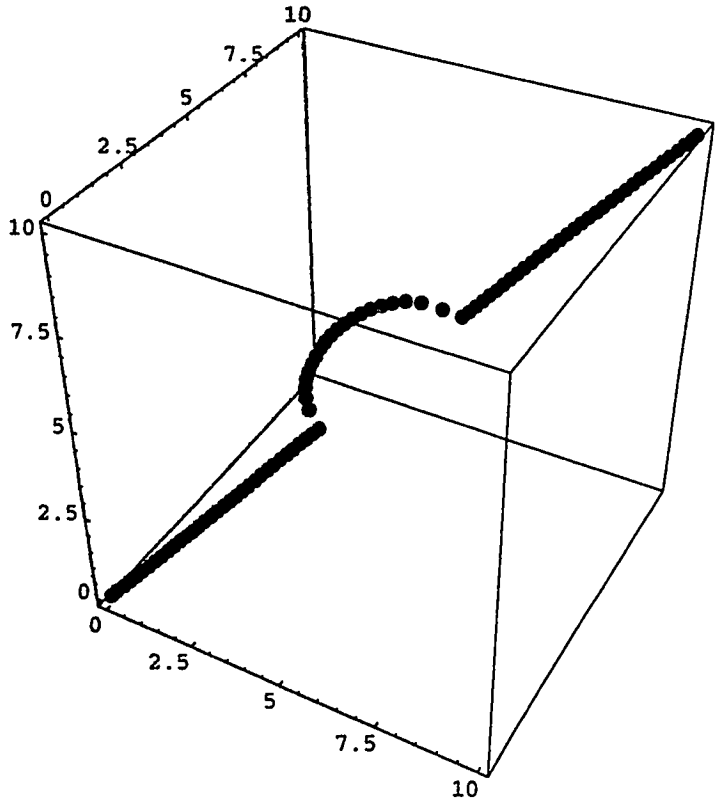


Figure 6.19: Path13

# Chapter 7

# SUMMARY AND CONCLUSIONS

In this thesis we have developed two collision detection algorithms for detecting the collision of three dimensional objects moving on an arbitrary path that can be parameterized as a function of a parameter $t$ (usually time). The algorithms can handle the case of objects moving with simultaneous translation and rotation.

The first algorithm detects collision of objects which are modelled as convex polyhedra defined as intersection of half-spaces which can be represented by a set of linear inequalities.

The second algorithm also detects collision of objects by modelling them as convex polyhedral sets, but defines these sets as convex hulls of finite number of vertices. This representation is easier to derive and its accuracy increases as the number of vertices used increase.

We have followed the *"multiple Interference"* detection methodology. This is one of the simplest approaches to the collision detection problem. In this approach, the position and orientation of the moving object(s) are calculated at discrete intervals of time (or any path parameter), and static interference detection is performed at each point. The time interval between successive checks is kept short to avoid missing a collision. However, if this time is made too short, the

computation time may become too long for practical purposes.

In our own approach to the collision detection problem, we used homogeneous transformation matrices to represent the positions and orientations of the objects at the discrete points in time and space. Using the above representations, the geometries of the objects are determined at each point in time, and static interference checks are performed using linear programming techniques. These came handy since the above representations give rise to linear systems, and efficient algorithms for linear programming are available.

The above simple procedure is repeated for all the discrete points on the path until the end.

In our implementation of the proposed collision detection algorithms, we have used a standard linear programming routine "LP" from the MATLAB Optimization Toolbox. The MATLAB environment provides suitable facilities to code the algorithms with little programming hardship typified by other programming languages such as Fortran, Pascal and C-language. Complex matrix manipulations that might involve hundreds of Fortran statements are performed by one statement of MATLAB.

Furthermore, the MATLAB environment allows the specifications of the objects, their motions and the time span of interest to be inputed from the screen interactively. Hence the codes have the features of other geometrical modelling systems. The only drawback in using MATLAB for implementation of the algorithms is that, its speed in executing complex "do loops" and calling other routines from different files is much slower than Fortran and C [93].

From the results of the simulations carried out in chapter 4, it has been shown that Algorithm I is more efficient than Algorithm II in computational time. However, both algorithms have linear computational complexities. But their efficiency lies heavily on the decisive choice of the time step $\Delta t$. Too large a value of $\Delta t$ will miss some collisions, and too small a value wastes a lot of

time. Furthermore, the algorithms can handle three dimensional objects translating and rotating along an arbitrary time parameterized path. And although the algorithms proceed to find all possible collisions between the potentially colliding objects, they can be modified to terminate after detecting the first collision.

Finally, it should be realized that the algorithms check for interference between the moving object of interest, and all the other objects at each discrete point in time on the path even though it is not necessary to do so, since objects away from the current location can be eliminated. This adds to the computational time of the algorithms. Therefore, a natural extension to this work should be to examine such a possibility. The modification of the algorithms to detect possible collisions for on-line application where the objects motions are uncertain should also be investigated.

We have also presented a methodology for local off-line path planning based on the method of potential functions for robotics application. It applies to static obstacles in the workspace whose presence is coded using a new artificial potential function. The new function possesses all the properties desired for an artificial potential function, the most important of which is freedom from local minima that can be created by addition with other potentials. All that is required to generate a potential field using the new function is an analytic description of the obstacles. For smooth objects such as spherical and ellipsoidal objects, this is readily available. Nonetheless, other convex objects can be approximately described using superquadric functions [47].

In addition to the path planning scheme, we have also proposed a new approach to path optimization that can be used with any potential or penalty function by parameterizing the path as a homotopy in the radius of an expanding sphere. Using simulations in two and three dimensions with a point mobile robot, we have demonstrated that a combination of the new

potential function for obstacle avoidance and the path optimization scheme generate a smooth continuous path from start to goal within a very short time. This is possible because the above formulation allows the planner the utilization of efficient optimization routines vis-a-vis the use of first order information provided by the gradient. We have solved a rich class of problems and we have shown representative solutions. Furthermore, we have shown that the potential function can be applied to the path planning of a rich class of nonpoint robots by transforming the workspace obstacles to configuration-space(CS) obstacles using an object growth algorithm.

Several motivations have led to the use of the potential field representation of the topological nature of free space. It can be used to obtain global representation of the space so that coarse planning can be done at the global level. Through an integration of our potential field-based path planning algorithm and our collision detection algorithms, an efficient global planner can be synthesized. It will not be difficult to use the path planner for global planning and the collision detector for local or fine planning.

As areas of future research, it is worth investigating the use of the new potential function for on-line obstacle avoidance using feedback control. We also intend to unify our approach to the path planning problem by considering both translations and rotations of the robot in three dimensions. The integration of the collision detection algorithms with the path planning algorithm into a global planner will also be an achievement. Furthermore, the use of the new potential function for motion planning in nonstationary environments and multiple (possibly interacting) robots in a workspace, should also be investigated. Applications to motion planning with uncertainty, holonomic and nonholonomic constraints will also be worthwhile.

# Appendix A

# PROGRAM 1

```
          PROGRAM FOR COLLISION DETECTION NO.1

      USING LINEAR INEQUALITIES REPRESENTATION OF THE OBJECTS

             THE CASE OF PURE TRANSLATION

*************************************************************************

n=input('number of objects=')

ts=input('initial time=')

tf=input('final time=')

h=input('time step=')

disp('initial orientation of the robot')

disp('in terms of Euler angles')

phi=input('roll angle=')

theta=input('pitch angle=')

psi=input('yaw angle=')

nx=cos(phi)*cos(theta);
```

```
ny=sin(phi)*cos(theta);

nz=-sin(theta);

sx=cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi);

sy=sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi);

sz=cos(theta)*sin(psi);

ax=cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi);

ay=sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi);

az=cos(theta)*cos(psi);

%************************************************************************

disp('input constraint matrices of the objects')

a=input('robot a=');b=input('b=');

a1=input('a1=');b1=input('b1=');a2=input('a2=');b2=input('b2=');

a3=input('a3=');b3=input('b3=');a4=input('a4=');b4=input('b4=');

a5=input('a5=');b5=input('b5=');a6=input('a6=');b6=input('b6=');

a7=input('a7=');b7=input('b7=');a8=input('a8=');b8=input('b8=');

a9=input('a9=');b9=input('b9=');a10=input('a10=');b10=input('b10=');

[ma1,na1]=size(a1);[ma2,na2]=size(a2);[ma3,na3]=size(a3);

[ma4,na4]=size(a4);[ma5,na5]=size(a5);[ma6,na6]=size(a6);

%******************MAIN LOOP*************************************************

t=ts;N=(tf-ts)/h +1

disp('define robot motion')

disp('input the coefficients of the polynomials px,py,pz up to third order')

cx=input('coefficients of px in descending order=')
```

```
cy=input('coefficients of py in descending order=')

cz=input('coefficients of pz in descending order=')

t1=clock;

for i=1:N

px=cx(4)+cx(3)*t+cx(2)*t^2+cx(1)*t^3;

py=cy(4)+cy(3)*t+cy(2)*t^2+cy(1)*t^3;

pz=cz(4)+cz(3)*t+cz(2)*t^2+cz(1)*t^3;

for j=1:n;

rr=[nx sx ax;ny sy ay;nz sz az];pr=[px;py;pz];

ar=a*rr';br=b+ar*pr;

if j==1 a0=a1;b0=b1;elseif j==2;a0=a2;b0=b2;elseif j==3;a0=a3;b0=b3;

elseif j==4;a0=a4;b0=b4;elseif j==5;a0=a5;b0=b5;elseif j==6;a0=a6;b0=b6;

elseif j==7;a0=a7;b0=b7;elseif j==8;a0=a8;b0=b8;elseif j==9;a0=a9;b0=b9;

elseif j==10;a0=a10;b0=b10;end;f=[1 0 0];

%********NEW SYSTEM OF INEQUALITIES AND SOLUTION*****************************

s=[ar;a0];d=[br;b0];disp('********Begin*******');disp(j);disp(t);

disp('collision point x =');

xc=lp(f,s,d,zeros(3,1));

disp(xc);

end;t=ts+i*h;end

t2=clock;etime(t2,t1)
```

# Appendix B

# PROGRAM 2

```
                PROGRAM FOR COLLISION DETECTION NO.2

          USING CONVEX HULLS REPRESENTATION OF  THE OBJECTS

                   THE CASE OF PURE TRANSLATION

*****************************************************************************

n=input('number of objects=')

ts=input('initial time=')

tf=input('final time=')

R=input('dimension of the problem=')

h=input('time step=')

disp('initial orientation of the robot')

disp('in terms of Euler angles')

phi=input('roll angle=')

theta=input('pitch angle=')

psi=input('yaw angle=')
```

```
nx=cos(phi)*cos(theta);

ny=sin(phi)*cos(theta);

nz=-sin(theta);

sx=cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi);

sy=sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi);

sz=cos(theta)*sin(psi);

ax=cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi);

ay=sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi);

az=cos(theta)*cos(psi);

%*************************************************************************

disp('input the vertices of the objects in homogeneous coordinates:')

o=input('o: robot=')

o1=input('o1= ');

o2=input('o2=');

o3=input('o3=');

o4=input('o4=');

o5=input('o5=');

o6=input('o6=');

o7=input('o7=');

o8=input('o8=');

o9=input('o9=');

o10=input('o10=');

%******************MAIN LOOP*************************************************
```

```
t=ts;N=(tf-ts)/h +1
disp('define robot motion')
disp('input the coefficients of the polynomials px,py,pz up to third order')
cx=input('coefficients of px in descending order=')
cy=input('coefficients of py in descending order=')
cz=input('coefficients of pz in descending order=')
t1=clock;
for i=1:N
px=cx(4)+cx(3)*t+cx(2)*t^2+cx(1)*t^3;
py=cy(4)+cy(3)*t+cy(2)*t^2+cy(1)*t^3;
pz=cz(4)+cz(3)*t+cz(2)*t^2+cz(1)*t^3;
for j=1:n;
Tr=[nx sx ax px;ny sy ay py;nz sz az pz;0 0 0 1];Or=Tr*o;
if j==1;o0=o1;elseif j==2;o0=o2;elseif j==3;o0=o3;elseif j==10;o0=o10;
elseif j==4;o0=o4;elseif j==5;o0=o5;elseif j==6;o0=o6;
elseif j==7;o0=o7;elseif j==8;o0=o8;elseif j==9;o0=o9;end;
[mo0,no0]=size(o0);[mo,no]=size(o);
%********NEW SYSTEM OF EQUATIONS AND SOLUTION*****************************
cr1=ones(1,no);cr2=zeros(1,no0);do1=ones(1,no0);do2=zeros(1,no);
if R==3;a=[Or(1:3,:) -o0(1:3,:);cr1 cr2;do2 do1];b=[0;0;0;1;1];
elseif R==2;a=[Or(1:2,:) -o0(1:2,:);cr1 cr2;do2 do1];b=[0;0;1;1];end;
[ma,na]=size(a);
vlb=zeros(na,1);f=[1 zeros(1,na-1)];disp('****begin****');disp(j);disp(t);
```

```
x=lp(f,a,b,vlb,[],[],ma);

disp('collision point')

xc=Or*x(1:no);disp(xc)

end;t=ts+i*h;end;t2=clock;

etime(t2,t1)
```

# Appendix C

# PROGRAM 3

```
                PROGRAM FOR COLLISION DETECTION No.3
      USING LINEAR INEQUALITIES REPRESENTATION OF THE OBJECTS
         THE CASE OF SIMULTANEOUS TRANSLATION AND ROTATION
*****************************************************************************
n=input('number of objects=')
ts=input('initial time=')
tf=input('final time=')
h=input('time step=')
R=input('dimension of the problem=')
disp('initial orientation of the robot')
disp('in terms of Euler angles')
phi0=input('roll angle=')
```

```
theta0=input('pitch angle=')

psi0=input('yaw angle=')

%*************************************************************************

disp('constraint matrices of the objects')

a=input('robot a=');b=input('b=');

a1=input('a1=');b1=input('b1=');a2=input('a2=');b2=input('b2=');

a3=input('a3=');b3=input('b3=');a4=input('a4=');b4=input('b4=');

a5=input('a5=');b5=input('b5=');a6=input('a6=');b6=input('b6=');

a7=input('a7=');b7=input('b7=');a8=input('a8=');b8=input('b8=');

a9=input('a9=');b9=input('b9=');a10=input('a10=');b10=input('b10=');

%******************MAINLOOP*****************************************************

t=ts;N=(tf-ts)/h +1

disp('define robot motion')

disp('input the coefficients of the polynomials px,py,pz up to third order')

cx=input('coefficients of px in descending order=')

cy=input('coefficients of py in descending order=')

cz=input('coefficients of pz in descending order=')

cphi=input('coefficient of roll=')

ctheta=input('coefficient of pitch=')

cpsi=input('coefficient  of yaw=')

t1=clock;

for i=1:N

phi=phi0+2*pi*t*cphi/(tf-ts);
```

```
theta=theta0+2*pi*t*ctheta/(tf-ts);

psi=psi0+2*pi*t*cpsi/(tf-ts);

px=cx(4)+cx(3)*t+cx(2)*t^2+cx(1)*t^3;

py=cy(4)+cy(3)*t+cy(2)*t^2+cy(1)*t^3;

pz=cz(4)+cz(3)*t+cz(2)*t^2+cz(1)*t^3;

nx=cos(phi)*cos(theta);

ny=sin(phi)*cos(theta);

nz=-sin(theta);

sx=cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi);

sy=sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi);

sz=cos(theta)*sin(psi);

ax=cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi);

ay=sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi);

az=cos(theta)*cos(psi);

rr=[nx sx ax;ny sy ay;nz sz az];pr=[px;py;pz];ppr=[ppr pr];

ar=a*rr';br=b+ar*pr;

for j=1:n;

if j==1 a0=a1;b0=b1;elseif j==2;a0=a2;b0=b2;elseif j==3;a0=a3;b0=b3;

elseif j==4;a0=a4;b0=b4;elseif j==5;a0=a5;b0=b5;elseif j==6;a0=a6;b0=b6;

elseif j==7;a0=a7;b0=b7;elseif j==8;a0=a8;b0=b8;elseif j==9;a0=a9;b0=b9;

elseif j==10;a0=a10;b0=b10;end;

%********NEW SYSTEM OF INEQUALITIES AND SOLUTION*******************************

if R==3;s=[ar;a0];d=[br;b0];f=[1 0 0];
```

```
elseif R==2;s=[ar(:,1:2);a0(:,1:2)];d=[br;b0];f=[1 0];end;

disp('********Begin*******');disp(j);disp(t);

disp('collision point x =');

%xc=relax1(s,d);

xc=lp(f,s,d,zeros(R,1));

disp(xc);end;t=ts+i*h;end

t2=clock;etime(t2,t1)
```

# Appendix D

# PROGRAM 4

```
          PROGRAM FOR COLLISION DETECTION NO.4
        USING CONVEX HULLS REPRESENTATION OF THE OBJECTS
        THE CASE OF SIMULTANEOUS TRANSLATION AND ROTATION
*************************************************************************
n=input('number of objects=')
ts=input('initial time=')
tf=input('final time=')
R=input('dimension of the problem=')
h=input('time step=')
disp('initial orientation of the robot')
disp('in terms of Euler angles')
phi0=input('roll angle=')
theta0=input('pitch angle=')
psi0=input('yaw angle=')
```

```
%*************************************************************************
disp('input the vertices of the objects in homogeneous coordinates:')
o=input('o: robot=')
o1=input('o1= ');
o2=input('o2=');
o3=input('o3=');
o4=input('o4=');
o5=input('o5=');
o6=input('o6=');
o7=input('o7=');
o8=input('o8=');
o9=input('o9=');
o10=input('o10=');
%******************MAIN LOOP*************************************************
t=ts;N=(tf-ts)/h +1
disp('define robot motion')
disp('input the coefficients of the polynomials px,py,pz up to third order')
cx=input('coefficients of px in descending order=')
cy=input('coefficients of py in descending order=')
cz=input('coefficients of pz in descending order=')
cphi=input('coefficient of roll=')
ctheta=input('coefficient of pitch=')
cpsi=input('coefficient of yaw=')
```

```
t1=clock;
for i=1:N
phi=phi0+cphi*2*pi*t/(tf-ts);
theta=theta0+ctheta*2*pi*t/(tf-ts);
psi=psi0+cpsi*2*pi*t/(tf-ts);
px=cx(4)+cx(3)*t+cx(2)*t^2+cx(1)*t^3;
py=cy(4)+cy(3)*t+cy(2)*t^2+cy(1)*t^3;
pz=cz(4)+cz(3)*t+cz(2)*t^2+cz(1)*t^3;
nx=cos(phi)*cos(theta);
ny=sin(phi)*cos(theta);
nz=-sin(theta);
sx=cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi);
sy=sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi);
sz=cos(theta)*sin(psi);
ax=cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi);
ay=sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi);
az=cos(theta)*cos(psi);
%for j=1:n;
Tr=[nx sx ax px;ny sy ay py;nz sz az pz;0 0 0 1];Or=Tr*o;
for j=1:n;
if j==1;o0=o1;elseif j==2;o0=o2;elseif j==3;o0=o3;elseif j==10;o0=o10;
elseif j==4;o0=o4;elseif j==5;o0=o5;elseif j==6;o0=o6;
elseif j==7;o0=o7;elseif j==8;o0=o8;elseif j==9;o0=o9;end;
```

```
[mo0,no0]=size(o0);[mo,no]=size(o);
%********NEW SYSTEM OF EQUATIONS AND SOLUTION*******************************
cr1=ones(1,no);cr2=zeros(1,no0);do1=ones(1,no0);do2=zeros(1,no);
if R==3;a=[Or(1:3,:) -o0(1:3,:);cr1 cr2;do2 do1];b=[0;0;0;1;1];
elseif R==2;a=[Or(1:2,:) -o0(1:2,:);cr1 cr2;do2 do1];b=[0;0;1;1];end;
[ma,na]=size(a);
vlb=zeros(na,1);f=[1 zeros(1,na-1)];disp('****begin****');disp(j);disp(t);
x=lp(f,a,b,vlb,[],[],ma);
disp('collision point')
xc=Or*x(1:no);disp(xc)
end;t=ts+i*h;end;t2=clock;
etime(t2,t1)
```

# Bibliography


[1] Agarwal, P. K. and M. Sharir, "Red-Blue Intersection Detection Algorithms, with Application to Motion Planning and Collision Detection," *SIAM J. of Comput.* Vol.19, No.2, pp.297-321, April,1990.

[2] Ahuja, N., R. T. Chien, R. Yen and N. Bridwell, "Interference Detection and Collision Avoidance Among Three Dimensional Objects," *First Annual National Conf. on Artif. Intell.*,Stanford, pp.44-48, Aug. 1980.

[3] Akgul, M. *Topics in Relaxation and ellipsoidal Methods.* Research Notes in Mathematics 97, Pitman Advanced Publishing Program, 1984.

[4] Al-Sultan, K. S., *"Nearest Point Problems: Theory and Algorithms"*, Ph.D Dissertation, University of Michigan, Ann Arbor, June 1990.

[5] Basta, R. A., R. Mehrotra , M. R. Varanasi, "Collision Detection for Planning Collision-free Motion of Two Robot Arms," *Proc. IEEE Conf. Robotics and Automation*, vol. 1,1988.

[6] Bazaraa M. S., Shetty C. M. *NonLinear Programming: Theory and Algorithms* John Wiley & Sons, USA. 1979.

[7] Bazaraa, M. S.,J. J. Jarvis., H. D.Shefali,*Linear Programming and Network Flows,* John Wiley Sons, 1990.

[8] Bobrow, J.E., "Optimal Robot Path Planning Using the Minimum-Time Criterion", *IEEE Trans. Robotics and Automation*, Vol.4, No.4, pp.443-450, August ,1988.

[9] Bonner, S. and R. B. Kelley "A Representation Scheme for Rapid 3-D Collision Detection," *Proc. IEEE Int. Symposium on Intelligent Control*, pp.320-325,1988.

[10] Borenstein, J. and Y. Koren (1989) High-Speed Obstacle Avoidance for Mobile Robots. *IEEE Int. Symp. on Intell Control*, pp.382-384.

[11] Boyse W.J. "Interference Detection Among Solids and Surfaces," *Comm. of the ACM*, Vol.22, No.1, pp.3-9 ,1979.

[12] Brooks, R. A., "Solving the Find-Path Problem by Good Representation of Free-Space," *IEEE Trans. Sys. Man. and Cybern.*, Vol.SMC-13, No.2, pp.190-197, March/Apr. 1983.

[13] Brooks R. A., "Planning Collision-Free Motions for Pick and Place Operations," *First Int. Symposium Robotics Research*, pp.5-37, 1993.

[14] Brooks, R. A. and T. Lozano-Perez, "A Subdivision Algorithm in Configuration Space for Find-Path With Rotation," *IEEE Trans. Sys. Man and Cybern.*,Vol.SMC-15, No.2, pp.224-233, Mar/Apr. 1985.

[15] Cameron, S. A. "A Study of the Clash Detection Problem in Robotics," *Proc. Int.Conf. of Robotics and Automation*, pp.488-493, 1985.

[16] Cameron, S. A. "Collision Detection by Four Dimensional Intersection Testing," *IEEE Trans. Robotics and Automation*, Vol. 6 No. 3, June 1990.

[17] Canny, J., "Collision Detection for Moving Polyhedra," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol.PAMI-8, pp.200-209, 1986.

[18] Canny, J. F. *The Complexity of Robot Motion Planning*, ACM Doctoral Dissertation Award ,MIT Press, 1987.

[19] Chien, R. T.,L. Zhang and B. Zhang, "Planning Collision-Free paths for Robotic Arm Among Obstacles," *IEEE Trans. Pattern Anal. and Machine Intell.*, Vol. PAMI-6, No.1, pp.91-96, January, 1984.

[20] Chin, F. and C. A. Wang, "Optimal Algorithm for the Intersection and Minimum Distance Problems Between Planar Polygons," *IEEE Trans. Computers*, Vol.32, pp.1203-1207, 1983.

[21] Craig J. J. *Introduction to Robotics Mechanics and Control*, Second Ed. Addison-Wesley Pub. Co. 1991.

[22] Crowly, J. L. "Navigation for an Intelligent Mobile Robot," *IEEE Trans, Robotics and Automation*, Vol.RA-1, No.1, pp.31-41, March, 1985.

[23] Culley, R.K.,and K. G. Kempf, "A Collision Detection Algorithm Based on Velocity and Distance Bound," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1064-1069, 1986.

[24] Elnagar, A. and A. Basu, "Heuristics for Local Path Planning", *IEEE Trans. Sys. Man and Cybern.*, Vol.23, No.2, pp.624-634, Mar/Apr. 1993

[25] Fang, S. C. and S. Puthenpura, *Linear Optimization and Extensions: Theory and Algorithms.* Prentice Hall 1993.

[26] Faverjon, B. and P. Tournassound, "A Local Based Approach for Path Planning of Manipulators with High Number of Degrees of Freedom," *Proc. IEEE Int. Conf. Robotics and Automation*, pp.1152-1159, 1987.

[27] Fu, K. S., R. C. Gonzalez and C. S. G. Lee, *Robotics: Control,Sensing, Vision and Intelligence.* Mcgraw-Hill Int. Edit. 1987.

[28] Fujimaru, K. and H. Samet, "A Hierachical Strategy for Path Plannig Among Moving Obstacles," *IEEE Trans. Robotics and Automation*, Vol.5, No.1, 61-69, Feb. 1989.

[29] Gallerini, R., "On Using LP to Collision Detection Between a Manipulator Arm and Surrounding Obstacles," *European Journal of Operational Research*, vol. 63,pp. 343-350, 1993.

[30] Gewali, L.P. ,S. Ntafos and I. G. Tollis, "Path Plannig in the Presence of Vertical Obstacles," *IEEE Trans Robotics and Automation*, Vol.6, No.3, pp.331-341, June 1990.

[31] Gilbert, E. G. and D. W. Johnson, "Distance Functions and Their Application to Robot Path Planning in the Resence of Obstacles. *IEEE Trans. Robotics and Automation*, Vol.RA-1, pp.21-30, March. 1985.

[32] Gilbert, E.G., D. W. Johnson and S. S. Keerthi, " A Fast Procedure for Computing the Distance Between Complex Objects in Three Dimensional Space," *IEEE Trans. Robotics and Automation*, Vol.RA-4, pp.193-203, 1988.

[33] Gilbert, E. G. and C. P. Foo, "Computing the Distance Between General Convex Objects in Three Dimensional Space," *IEEE Trans. Robotics and Automation*, Vol.6, No.1, Feb. 1990.

[34] Gilbert, E.G., and S. M. Hong, " New Algorithm for Detecting the Collision of Moving Objects," *Proc. IEEE Conf. Robotics and Automation*, pp.8-14, 1989.

[35] Gouzenes, L., "Strategies for Solving Collision-Free Trajectories Problems for Mobile and Manipulator Robots. *Int. Journal of Robotics Research*, Vol.3, No.4 pp.51-65, 1984.

[36] Hashimoto, H., Y. Kunii, F. Harashima, V. I. Utkin and S. V. Drakunov, "Obstacle Avoidance Control in Multi-Dimensional Space Using Sliding Mode," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp.697-702, July, 1992.

[37] Hayward, V. "Fast Collision Detection Scheme by Recursive Decomposition of a Manipulator Workspace," *Proc. IEEE Int. Conf. Robotics and Automation*, vols. 1-3,pp.1044-1049, April 1986.

[38] Hwang, Y. K. and N. Ahuja, "A potential Field Approach to Path Planning. *IEEE Trans. Robotics and Automation*, Vol.8, No.1, pp.23-32, February, 1992.

[39] Jagannathan, S., F. L. Lewis and K. Liu, "Modelling, Control and Obstacle Avoidance With an Onboard Manipulator," *Proc. Int. Symposium on Intell. Control*, pp.196-201, August, 1993.

[40] Jia, T. and M. L. Amirouche, "Natural Dynamics of Robot Manipulators in the Presence of Obstacles," *ASME Trans.Dynamic Systems Meas. and Control* Vol.113, pp.170-174, March, 1991.

[41] Johnson, D. W. and E. G. Gilbert, "Minimum Time Robot Path Planning in the Presence of Obstacles," *Proc. IEEE Conf. Decision and Control*, pp.1748-1753, Dec.1985.

[42] Johnson, D. W. The Optimization of Robot Motion in the Presence of Obstacles, *Ph.D. Dissertation.* University of Michigan, 1987.

[43] Kambhampati, S. K. and L. S. Davis, "Multiresolution Path Planning for a Mobile Robot," *IEEE Trans. Robotics and Automation*, Vol.RA-2, No.1, pp.135-145, Sept. 1986.

[44] Kawabe, S., A. Okano, K. Shimada, "Collision Detection Among Moving Objects in Simulation," *Proc. 4th Int. Symp. on Robotics Research*, pp.489-496, MIT Press, 1988.

[45] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *Int. Journal of Robotics Research*, Vol.5, No.1, pp.90-98, 1986.

[46] Kheradpir, S. and J. S. Thorp, "Real-Time Control of Robot Manipulators in the Presence of Obstacles," *IEEE Trans. Robotics and Automation*, Vol.4, No.6, pp.687-698, Dec. 1988.

[47] Khosla, P. and R. Volpe (1988) Superquadric Artificial Potentials for Obstacle Avoidance and Approach. *Proc. IEEE Int. Conf, Robotics and Automation*, Vol.3, pp.1778-1784.

[48] Khouri, J. and K. A. Stelson (1989) An Efficient Algorithm for Shortest Path in Three Dimensions with Polyhedral Obstacles. *ASME Trans. Dyn. Systems Meas. and Control* Vol.111, pp.433-436 Sept.

[49] Kim, B. K. and K. G. Shin (1985) Minimum-Time Path Planning for Robot Arms and Their Dynamics. *IEEE Trans. Sys. Man and Cybern.*, Vol.SMC-15, No.2, pp.213-223, Mar/Apr.

[50] Kim, J. O. and P. K. Khosla (1992) Real-Time Obstacle Avoidance Using Harmonic Potential Functions. *IEEE Trans. Robotics and Automation*, Vol.8, No.3, pp.338-349, June.

[51] Kyriakopoulos, K. J., and G. N. Saridis (1992) Distance Estimation and Collision Prediction for On-line Robotic Motion Planning. *Automatica* vol.28 ,No.2,pp. 389- 394

[52] Kyriakopoulos, K. J. and G. N. Saridas (1993) An Integrated Collision Prediction and Avoidance Scheme for Mobile Robots in Non-stationary Environment. *Automatica*, Vol. 29, pp.309-322,

[53] Latombe J.C. (1991) *Robot Motion Planning.* Kluwer Academic Publishers, Boston.

[54] Laurini, R. and D. Thomson, *Fundamentals of Spatial Information Systems.* Academic Press, A.P.I.C. Series,1992.

[55] Lozano-Perez, T.(1981) Automatic Planning of Manipulator Transfer Movements. *IEEE Trans. Sys. Man and Cybern.* Vol.SMC-11, No.10, pp.681-697, Oct.

[56] Lozano-Perez, T. and M. A. Wesley (1979) An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles. *Comm. of the ACM*, Vol.22, No.10, pp.560-570, October.

[57] Lozano-Perez, T (1983) Spatial Planning: A Configuration Space Approach", *IEEE Trans. on Computers*, Vol.C-32, No.2, pp.108-119, February,

[58] Lozano-Perez, T. (1987) A Simple Motion-Planning Algorithm for General Robot Manipulators. *IEEE Trans. Robotics and Automation*, Vol.RA-3, No.3, pp.224-237, June.

[59] Lozano-Perez T., M. T. Mason and R. H. Taylor (1983) Automatic Synthesis of Fine-Motion Strategies for Robots *First Int. Symposium Robotics Research*, pp. 100-110.

[60] Luh, J. Y. S. (1984) A Scheme for Collision Avoidance with Minimum Distance Traveling for Industrial Robots. *Jounal of Robotics Systems*, Vol.1, No.1, pp.5-26

[61] Luh, J. Y. S. and C. E. Campbell (Jr.) (1984) Minimum Distance Collision-Free Path Planning for Industrial Robots with a Prismatic Joint. *IEEE Trans. Automatic Control* Vol.AC-29, No.8, pp.675-680, Aug.

[62] Lumelsky, V. J. "On fast Computation of Distance Between Line Segments," *Inf. Proc. Lett.*,Vol 21, pp.55-61, 1985.

[63] Lumelsky, V. J. and A. A. Stepanov (1986) Dynamic Path Planning for an Automaton With Limited Information on the Environment *IEEE Trans. Automatic Control*, Vol.AC-31, No.11, pp.1058-1063, Nov.

[64] Lumelsky, V. J., S. Mukhopadhyay and K. Sun (1990) Dynamic Path Planning in Sensor-Based Terrain Acquisition. *IEEE Trans. Robotics and Automation*, Vol.6,No.4, pp.462-472, Aug.

[65] Meyer, W., "Distances Between Boxes: Application to Collision Detection and Clipping," *Proc. Int. Conf. on Robotics and Automation*, pp.597-602, 1986.

[66] Ozaki, H., A. Mohri, and M. Takata (1982) Planning of Collision Free Movement of a Manipulator Considering its Body Space. *Trans. of the Soc. of Instrument and Control Engineers(Japan)*, Vol.18, part 9, pp.942-949.

[67] Pavlov, V. V. and A. N. Voronin (1984) The Method of Potential Functions for Coding Constraints of the External Space in an Intelligent Mobile Robot *Soviet Automatic Control*, Vol.17, pp.45-51, Nov-Dec.

[68] Petrov, A. A. and I. M. Sirota (1983), "Obstacle Avoidance by a Robot Under Limited Information About the Environment. *Automation and Remote Control*, pp.431-439, 1983.

[69] Phillip, J. M.(1991) *Introduction to Robotics,* University of Wollongong, Australia, Addison-Wesley Pub. Co.

[70] Preparata, F. P. and M. I. Shamos,*Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[71] Rimon, E. and D. E. Koditschek (1988) Exact Robot Navigation Using Cost Functions: The Case of Distinct Spherical Boundaries in $E^n$", *IEEE Int. Conf. Robotics and Automation*, Vol.3, pp.1791-1796.

[72] Rimon, E. and D. E. Koditschek (1992) Exact Robot Navigation Using Artificial Potential Functions. *IEEE Trans. Robotics and Automation*, Vol.8, No.5, pp.501-518, October.

[73] Raymond, S. R. and A. S. El-Gizawy (1992) "Minimum time Collision-Free Path Planning for Robotic Assembly. *Int. J. Prod. Res.*, Vol.30, No.6, pp.1301-1312

[74] Schwartz, J. T., "Finding the Minimum Distance Between Two Convex Polygons," *Inf. Procc. Lett.*, vol.13, No.s 4,5, 1981.

[75] Schwartz, J. T. and M. Sharir (1983) On Piano Movers Problem 1 : The Case of a Two-Dimensional Rigid Polygonal Body Amidst Polygonal barriers *Comm. Pure and Applied Math.*, Vol.36, pp.345-398.

[76] Schweikard, Achim, "Polynomial Time Collision Detection for Manipulator Paths Specified by Joint Motions," *IEEE Trans. Robotics and Automation*, vol.7 No.6,pp.865-869, 1991.

[77] Seshadri, C. and A. Ghosh (1993) Optimum Path Planning for Robotic Manipulators Amid Static and Dynamic Obstacles. *IEEE Trans. Sys. Man. and Cybern.* Vol.23, No.2, pp.576-584, Mar/Apr.

[78] Shigematsu, Y., Y. Kakazu and N. Okino , "Interference Detection Algorithm by Simplex Method," *Journal of Japanese Society of Precis. Eng (Japan)*, Vol. 49 Part 11, pp.1561-1566,1983.

[79] Shiller, Z. and S. Dubowsky (1989) Robot Path Planning with Obstacle, Actuator, Gripper and Pay-load Constraints. *Int. Journal of Robotics Research*, Vol.8, No.6, Dec.

[80] Shiller, Z. and Y. Gwo (1991) Dynamic Motion Planning of Autonomous Vehicles. *IEEE Trans. Robotics and Automation*, Vol.7, No.2, April

[81] Shin, K. G. and N. D. McKay (1985) Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints. *IEEE Trans Automatic Control*, Vol.AC-30, No.6, pp.531-541, June.

[82] Shin, K. G. and N. D. Mckay (1986) Selection of Near-Minimum time Geometric Path for Robotic Manipulators. *IEEE Trans. Automatic Control*, Vol.AC-31, No.6, pp.501-511, June.

[83] Suh, S. and K. G. Shin (1988) A Variational Dynamic Programming Approach to Robot-Path Planning With a Distance-Safety Criterion. *IEEE Trans. Robotics and Automation*, Vol.4, No.3, pp.334-349, June.

[84] Tilove, R. B. "A Null Object Detection Algorithm for Constructive Solid Geometry *Communications of the ACM*, Vol. 27,No.7, July 1984.

[85] Tilove, R. B (1990) Local Obstacle Avoidance for Mobile Robots Based on the method of Artificial Potential Functions. *IEEE Int. Conf. on Robotics and Automation*, pp.566-571, May, 1990.

[86] Tournassound, P. (1988) Motion Planning for a Mobile Robot with Kinematic Constraint. *IEEE Int. Conf. Robotics and Automation*, pp.1785-1790, 1988.

[87] Uchiki, T, T. Ohashi and M. Tokoro, "Collision Detection in Motion Simulation," *Computer & Graphics*, Vol. 7, pp.285-293, 1983.

[88] Udupa, S. M. (1977) Collision Detection and Avoidance in Computer Controlled Manipulators. *Proc. 5th Int. Joint Conf. on Artificial Intell.*, Boston, vol.2, pp.737-748.

[89] Volpe, R. and P. Khosla (1987) Artificial Potentials with Elliptical Isopotential Contours for Obstacle Avoidance. *Proc. 26th IEEE Conf. Dec. and Control*, pp.180-185, Dec.

[90] Warren, C. W.(1989) Global Path Planning Using Artificial Potential Fields. *Proc. IEEE Int. Conf. Robotics and Automation*, pp.316-321, May, 1989.

[91] Wolfe, P. "Finding the Nearest Point in a Polytope," *Mathematical Programming*, Vol.11, pp.128-149, 1976.

[92] Wong, E. K. and K. S. Fu (1986) A Hierachical Orthogonal Space Approach to Three-Dimensional Path Planning *IEEE Trans. Robotics and Automation*, Vol.RA-2, No.1, pp.61-69, March.

[93] 386 MATLAB User manual(1989), MATHWORKS Inc., MA, USA.

[94] MATLAB Optimization Toolbox manual(1992), MATHWORKS Inc., MA, USA.

# Vita

- Mohammad Dikko S. Aliyu
- Born in Kano, Nigeria 1966.
- Studied B.Eng (Electrical) A.B.U. Zaria, Nigeria, 1988.