NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.
MOTION PLANNING FOR MANIPULATORS:
A NOVEL APPROACH
WITHIN A SEQUENTIAL FRAMEWORK

by

Xinyu Zhu
M.ENG. Tsinghua University, 1986
B.ENG. Tsinghua University, 1984

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in the School
of
Engineering Science

© Xinyu Zhu 1994
SIMON FRASER UNIVERSITY
December 1994

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without the permission of the author.
THE AUTHOR HAS GRANTED AN IRREVOCABLE NON-EXCLUSIVE LICENCE ALLOWING THE NATIONAL LIBRARY OF CANADA TO REPRODUCE, LOAN, DISTRIBUTE OR SELL COPIES OF HIS/HER THESIS BY ANY MEANS AND IN ANY FORM OR FORMAT, MAKING THIS THESIS AVAILABLE TO INTERESTED PERSONS.

THE AUTHOR RETAINS OWNERSHIP OF THE COPYRIGHT IN HIS/HER THESIS. NEITHER THE THESIS NOR SUBSTANTIAL EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT HIS/HER PERMISSION.

L'AUTEUR A ACCORDE UNE LICENCE IRREVOCABLE ET NON EXCLUSIVE PERMETTANT A LA BIBLIOTHEQUE NATIONALE DU CANADA DE REPRODUIRE, PRETER, DISTRIBUER OU VENDRE DES COPIES DE SA THESE DE QUELQUE MANIERE ET SOUS QUELQUE FORME QUE CE SOIT POUR METTRE DES EXEMPLAIRES DE CETTE THESE A LA DISPOSITION DES PERSONNE INTERESSEES.

L'AUTEUR CONSERVE LA PROPRIETE DU DROIT D'AUTEUR QUI PROTEGE SA THESE. NI LA THESE NI DES EXTRAITS SUBSTANTIELS DE CELLE-CI NE DOIVENT ETRE IMPRIMES OU AUTREMENT REPRODUITS SANS SON AUTORISATION.
APPROVAL

Name: Xinyu Zhu
Degree: Master of Applied Science
Title of thesis: Motion Planning For Manipulators: A Novel Approach within Sequential Framework

Examining Committee:

Chairman: Dr. Jacques Vaisey

Dr. Kamal Gupta
Senior Supervisor

Dr. Shahram Payandeh
Supervisor

Dr. Binay Bhattacharya
School of Computing Science, SFU
Supervisor

Dr. William Gruver
Examiner

Date Approved: 22/Dec/94
PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

"Motion Planning For Manipulators: A Novel Approach Within A Sequential Framework"

Author:

(signature)

Xinyu Zhu
(name)

October 20, 1994
(date)
Abstract

This thesis presents a novel approach within a sequential framework to develop practical motion planners for manipulators with many degrees of freedom (DOFs). The sequential framework is a paradigm for developing motion planners (Gupta 1990; Gupta and Guo 1994).

The essence of the sequential framework is to exploit the serial structure of manipulators and decompose the n-dimensional problem of planning collision-free motions for an n-DOF manipulator into a sequence of smaller \((m_j + 1)\)-dimensional sub-problems, \(m_1 + m_2 + \ldots + m_s = n\). The \(j\)th sub-problem corresponds to planning the motion of a sub-group of \(m_j\) links (DOFs) along a given path. Backtracking is used if no path exists for a given sub-problem. In this thesis, each sub-problem, a 2-dimensional motion planning problem, is solved by using numerical potential fields defined over bitmap-based representations. Furthermore, a systematic and efficient backtracking mechanism based on a novel notion of virtual forbidden regions in subspaces is presented. This approach leads to more efficient and robust motion planners than a previous implementation that used polygonal representations and visibility graphs (Gupta and Guo 1992).

The new approach has been successfully applied in both 2 and 3 dimensional workspaces. Experiments have been conducted for manipulators with up to 8 degrees of freedom among randomly or manually placed obstacles. Although it is not complete, the planner never failed for these examples in hundreds of simulations, and at most three backtracking levels were needed. For a 7-DOF manipulator in 3-dimensional workspaces, the average run-time is about 8 minutes. These empirical results clearly show that the approach is efficient and practically useful.
Acknowledgements

I would like to express my appreciation to my senior supervisor Dr. Kamal Gupta for help and advice concerning not only my thesis, but many other areas of my academic career as well.

I would like to thank Brigitte Rabold, Susan Stevenson, Karen Dias, Zhenping Guo, Hezhi Fan and others for their valuable help and discussions on my thesis.

My special thanks to my wife and my parents, for encouraging me and supporting me in my academic endeavors.
# Contents

Abstract .................................................................................................................. iii

Acknowledgements .................................................................................................. iv

List of Tables .......................................................................................................... x

List of Figures ......................................................................................................... xiv

List of Abbreviations ............................................................................................... xv

1 Introduction ........................................................................................................... 1

1.1 The Motion Planning Problem: Definition ...................................................... 2

1.2 Difficulties in Developing Practical Motion Planners ..................................... 3

1.3 Motivation ......................................................................................................... 4

1.4 An Overview of the Sequential Framework .................................................. 5

1.5 Previous Implementation ................................................................................. 6

1.6 A Novel Approach within a Sequential Framework ...................................... 8

1.7 Organization of the Thesis ............................................................................. 11

2 Related Work ...................................................................................................... 12
2.1 Theoretical Aspects .................................................. 13
2.2 Practical Motion Planners ............................................. 14
2.3 Explicit Computation of Configuration Space Based Planners ....... 15
  2.3.1 Lozano-Pérez's C-Space Approach ................................ 16
  2.3.2 Octree Representation Approaches .............................. 17
  2.3.3 Enumeration Approach ........................................... 17
2.4 Potential Field Based Motion Planners ............................... 18
  2.4.1 Randomized Path Planner ....................................... 19
2.5 Multi-Level Based Planners .......................................... 20
  2.5.1 Global-local Planner ........................................... 20
  2.5.2 SANDROS Planner ............................................... 22
2.6 Parallel Implementations ............................................. 23
  2.6.1 Parallel Computing of C-space .................................. 23
  2.6.2 Genetic Approaches ............................................. 23

3 Solving the Single Link Problem ....................................... 25
  3.1 Terminology and Notations ......................................... 25
  3.2 Kinematics .......................................................... 26
  3.3 A New Bitmap-Based Approach ..................................... 27
  3.4 Building Forbidden Regions in the \( t_i \times q_i \) Space: The \( t_i \times q_i \) Bitmap ... 27
    3.4.1 The \( t_i \times q_i \) Bitmap ..................................... 28
    3.4.2 Collision Detection in 2-Dimensional Workspace ............... 28
    3.4.3 Collision Detection in 3-Dimensional Workspace ............... 32
  3.5 Numerical Potential Fields and Search in the \( t_i \times q_i \) Space Bitmap ... 35
3.5.1 Build the \( L^1 \) Distance Map ........................................ 35
3.5.2 Numerical Potential Fields ........................................ 36
3.5.3 Gradient Search .................................................. 39
3.5.4 Forward and Backward Extensions ............................... 42
3.5.5 Reparameterization of the \( t_i \times q_i \) path .................... 42

4 Backtracking Strategy ................................................... 44
4.1 Determining Block and Virtual Forbidden Regions .................. 44
4.2 Algorithmic Description of the Planner ............................... 52
4.3 Empirical Complexity Analysis ....................................... 54

5 Experimental Results and Planner Evaluation .......................... 58
5.1 Examples on Several Manipulators .................................... 58
5.1.1 A Simple 4-DOF Arm in 3-D Workspace ........................... 58
5.1.2 A 6-DOF Arm in 2-D Workspace ................................... 61
5.1.3 An 8-DOF Arm in 2-D Workspace ................................... 62
5.1.4 A 7-DOF Example in 3-D Workspace ............................... 66
5.2 Experiments ............................................................ 68

6 Conclusions ...................................................................... 78
6.1 Summary ..................................................................... 78
6.2 Future Work .................................................................. 79

A Planner System and Representations .................................... 82
A.1 Link Coordinate Systems ............................................... 82
A.2 Planner Data Structures ................................................ 85

vii
A.2.1 Data Structures for a Robot Arm ......................... 85
A.2.2 Data Structures for $t \times q$ Space and Paths ............. 88

References ........................................................................ 92
List of Tables

5.1 Detailed run times for a 4-DOF arm. ............................................. 61
5.2 Detailed run times for a 7-DOF manipulator. ................................. 71
5.3 Examples of run times for the four link manipulator arm, where 10 obstacles were randomly placed in the 2-D workspace. The maximum backtracking level was set at 2. In column 3, \( x(i) \) denotes that planner backtracked \( x \) number of times for link \( i \). ............................................. 73
5.4 Numerical results for the four link manipulator, where obstacles were randomly positioned in its environment, and the number of obstacles was 10. .................................................... 73
5.5 Examples of run times for the six link manipulator, where 6 obstacles were randomly placed. The maximum backtracking level was set at 3. In column 3, \( x(i) \) denotes that the planner backtracked \( x \) number of times for link \( i \). .................................................... 74
5.6 Numerical results for the six-link manipulator, where obstacles were randomly positioned in its environment. ................................. 74
5.7 Numerical results for the four link manipulator with 10 fixed obstacles. 74
5.8 Examples of run times for a 7-DOF manipulator in 3-dimensional workspaces. .................................................... 75
5.9 Average run times for the 7-DOF manipulator in 3-dimensional workspaces. .................................................... 76
5.10 Numerical results for a V-Graph based planner for a six link manipulator in 3-dimensional workspace. Obstacles were randomly placed in its environment, and the number of obstacles was varied randomly between 4 and 10.
List of Figures

1.1 An overview of the sequential framework, where each link is a subgroup. $\text{Max\_Level}$ is the maximum level of backtracking and is a user determined parameter. $B\_Level$ is the current level of backtracking. . . 7

1.2 An example of planned motion sequences for a 7-DOF STM xarm. . . 10

1.3 An example of planned motion sequences for an 8-DOF manipulator in a 2-D workspace. ................................................................. 11

2.1 C-space obstacles and path for a simple 2-DOF manipulator. (a) A manipulator with two revolute joints and its 5-obstacle workspace. (b) The corresponding C-space obstacles (the black regions) and C-space path. Where the 6 numbers in both figures correspond to 6 configurations in both spaces. .......................................................... 15

2.2 A C-space potential field that will not aid the random search. (a) 2-link arm in its initial and final position. (b) C-space potential. ........... 21

3.1 An example of a 4-dof arm in its 2-dimensional environment: (a) initial position and (b) final position. ................................. 29

3.2 Examples of two $t \times q$ forbidden region maps, where the bright area is free. ................................................................. 29

3.3 A hierarchical model for several obstacles, where a dotted-line parallelepiped object encloses three simple objects. ...................... 33
3.4 Free range prediction. The distance and radius of a link is used to estimated a free range $D/L$ (Note the radius in the figure is an approximation).

3.5 Two $t \times q$ distance maps. They correspond to the $t \times q$ spaces in Figure 3.2. Brighter area indicates larger distance values and the completely dark areas are forbidden.

3.6 A Voronoi diagram for 3 objects in a square.

3.7 Examples of Voronoi diagrams in the $t \times q$ spaces. They are generated for the $t \times q$ spaces in Figure 3.2. Many of the small branches of diagrams are caused by uneven boundaries of the $t \times q$ forbidden regions.

3.8 Two examples of the $t \times q$ potential fields. Brighter area represents larger potential values, the complete dark areas are the forbidden regions, and the half-dark areas are free but without potential.

3.9 The bitmaps for a 4-DOF planar arm. The $t_2 \times q_2$ space and path are at left figure, the $t_3 \times q_3$ is at right. Note that there is no path for link 3.

4.1 Examples of the “block”s in the $t \times q$ space.

4.2 Inserting virtual forbidden region.

4.3 Inserting more virtual forbidden regions.

4.4 The inserted virtual forbidden region in the $t_{i-1} \times q_{i-1}$ space is equivalent to a vertical block strip in the $t_i \times q_i$ space.

4.5 Determining $t_i^{block}$ in the $t_i \times q_i$ space. (a) a conical region is swept along the straight line from start to goal, (b) search is limited to a diamond shaped region.

4.6 Backtracking in the $t \times q$ spaces, (a) and (b) are the bitmaps and paths of $t_2 \times q_2$ and $t_3 \times q_3$ before backtracking, (c) and (d) are the bitmaps and paths of $t_2 \times q_2$ and $t_3 \times q_3$ after backtracking.
5.1 An example of the planned motion sequences for a 4-DOF manipulator

5.2 The $t \times q$ spaces for a 4-DOF manipulator. The $t_1 \times q_1$ to $t_4 \times q_4$ spaces are arranged from left to right. The sizes are $180 \times 180$, $180 \times 180$, $270 \times 180$, and $520 \times 180$ respectively.

5.3 An example of a 6-DOF manipulator. The initial (left configuration) and final (right configuration) positions workspace for an example of a 6-DOF manipulator in the 2-D workspace.

5.4 The continuous motion for a 6-DOF manipulator in a 2-D workspace.

5.5 The $t \times q$ spaces of a 6-DOF manipulator. The $t_1 \times q_1$ to $t_6 \times q_6$ spaces are from top to bottom and from left to right. The sizes are $360 \times 360$, $360 \times 360$, $1203 \times 360$, $1876 \times 360$, $2701 \times 360$, and $3137 \times 360$ respectively.

5.6 A complicated path in $t_5 \times q_5$ space.

5.7 An example of an 8-DOF manipulator in a 2-D workspace. The start is the left configuration and goal is the right configuration.

5.8 The continuous motion for an 8-DOF manipulator in 2-D workspace.

5.9 The $t \times q$ spaces of an 8-DOF manipulator. The $t_1 \times q_1$ to $t_8 \times q_8$ spaces are from top to bottom and from left to right. The sizes are $360 \times 360$, $360 \times 360$, $847 \times 360$, $1952 \times 360$, $2179 \times 360$, $2917 \times 360$, $3499 \times 360$, and $3023 \times 360$ respectively.

5.10 The SPDM testbed manipulator (STM).

5.11 An example of the planned motion sequences for a 7-DOF manipulator.

(Continued in next figure.)

5.12 An example of the planned motion sequences for a 7-DOF manipulator.

(Continuing of last figure.)
5.13 The $t \times q$ spaces of a 7-DOFs manipulator. The $t_1 \times q_1$ to $t_7 \times q_7$ spaces are from top to bottom and from left to right. The sizes are $180 \times 180$, $180 \times 180$, $268 \times 180$, $288 \times 180$, $509 \times 180$, $540 \times 180$, and $716 \times 180$ respectively.

A.1 Link coordinate systems and its parameters. ........................................... 83
A.2 Robot Coordinate Systems. ................................................................. 84
A.3 The relation between two link paths. .................................................. 90

xiv
List of Abbreviations

2-D 2-Dimensional
3-D 3-Dimensional
C-space Configuration Space
C-obstacle Configuration Space Obstacle
D-H Denavit-Hartenberg (Representation)
DOF Degree Of Freedom
SPDM Special Purpose Dextrous Manipulator
STM SPDM Testbed Manipulator
V-Graph Visibility Graph
Chapter 1

Introduction

Until recently, most industrial manipulators were employed for pre-programmed, repetitious tasks. With a pressing requirement for robots in service applications, the demand for more autonomous and intelligent robotic systems is increasing. Future generations of robots will be equipped with intelligent capacities, such as perception, planning, reasoning and learning.

An intelligent robotic system may observe the situation of the working environment by means of various sensing devices, build the conceptual world model from the raw observative data, make decisions itself according to certain requirements and rules, plan a sequence of motions for the manipulator, and finally command the low level control mechanism to accomplish the required task.

Consider a future automated assembly line in which mobile robots pick up parts and deliver them to assembly manipulators. The mobile robots must find their way to the parts, pick them up, move to the assembly manipulators, and pass them to the assembling manipulators. The assembling manipulators then carry the parts to their assembly positions and then fix them. Each of these motions have to be carefully planned without any collision with objects and other robots in the workspace. The aim of motion planning is to automatically plan a collision-free sequence of a manipulator’s motion with certain constraints. With the capacity of motion planning, abstract task descriptions, such as “Grasp part A” and “move part A to position B”, could be issued by a human operator or an intelligent task decomposing system. The detail of the
CHAPTER 1. INTRODUCTION

motion sequence is planned automatically instead of using the human-programmed motion currently employed in industry.

This motion planning problem has been studied for years. A number of variations of the problem exists in past research, such as the motion of a free-flying object with no constraints (Schwartz and Sharir 1983a; Schwartz and Sharir 1983b; Madilla 1986), motion with moving obstacles (Gupta and Zucker 1986; Gupta and Zucker 1987), or multi-manipulator coordination or multi-objects moving problem (Schwartz and Sharir 1983c; Erdmann and Lozano-Pérez 1986; Lee and Lee 1987; Roach and Boaz 1985).

This thesis concentrates on the motion planning problem for an articulated manipulator arm. The manipulator arm can be modeled as a chain of rigid links interconnected by joints (either prismatic or revolute) with one end of the chain fixed in a workspace. A novel approach for planning motions for such a manipulator with many degrees of freedom (DOFs) developed in (Gupta and Zhu 1993; Gupta and Zhu 1994) has been implemented. The following sections will define the motion problem studied in this thesis project, give a brief description of the approach, and emphasize the contributions of this project.

1.1 The Motion Planning Problem: Definition

The motion planning problem discussed in this thesis is defined similar to (Latombe 1991) and is briefly described as follows. Let $A$ be a manipulator arm consisting of a set of $n+1$ rigid links that are articulated to each other by $n$ prismatic or revolute joints. Suppose further that $A$ is operating in a workspace amidst a set of static obstacles whose geometry is known to the robot system in advance.

The motion planning problem for $A$ is

**Given a start configuration $q_s$ and a desired goal configuration $q_g$ of $A$, determine whether there exists a continuous collision-free motion of $A$ from $q_s$ to $q_g$, and, if so, plan such a motion.**
CHAPTER 1. INTRODUCTION

The manipulator arms are assumed to be articulated chains made up of a series of rigid links connected by one-degree-of-freedom joints, thus the manipulator possesses a total of \( n \) degrees of freedom, i.e., each position of \( A \) can be specified by \( n \) real parameters. The joint motions are either rotational or translational\(^1\). Such manipulators are also called serial manipulators. The terms manipulator arm, manipulator or arm used throughout the thesis will refer to serial manipulators unless otherwise specified. The positions of all links of the rigid robot arm are completely specified by the values of the joint parameters. A set of \( n \) such parameters that uniquely specifies the position of every link of the arm is called a configuration, and the space defined by those parameters is called the configuration space (C-space). The set of configurations that give rise to collisions in workspace is referred to as C-space obstacles or briefly C-obstacles. For a manipulator arm with \( n \) degrees of freedom, its joint space is an \( n \)-dimensional C-space (Lozano-Pérez and Wesley 1979; Lozano-Pérez 1983; Lozano-Pérez 1987). The C-space is the space that represents all possible motions of a manipulator. Once such a representation has been obtained, the actual motion planning problem is equivalent to searching a curve from the start configuration to the goal configuration in an \( n \)-dimensional C-space.

1.2 Difficulties in Developing Practical Motion Planners

At least 6 degrees of freedom are required to determine an arbitrary position and orientation for the end-effector of a manipulator. Therefore most manipulators (for example, the 6-DOF PUMA-560 and the 7-DOF SPDM space arm) have 6 or more degrees of freedom, and lead to high dimensionality of the configuration space. Clearly, dealing with an \( n \)-dimensional space is difficult. Most of the approaches developed for motion planning implicitly or explicitly compute the C-space of the manipulator and then search it for a path. The high dimensionality of C-space gives rise to a major problem with these approaches — the configuration space obstacles are highly complex to compute and represent analytically. A number of researchers (Faverjon

\(^1\)The implementation in this thesis is for rotational joints only, but it is easy to include translational joints.
1984; Lozano-Pérez 1987; Faverjon and Tournassoud 1987; Kondo 1991) use discrete representations of the configuration space. However, the memory requirement and the computational burden of the n-dimensional C-space obstacles are so heavy that it is impractical to build them using full resolution. For example, for a 6-DOF all-revolute-joint manipulator, at a resolution of 2 degree (angular), the memory required is in the order of \(180^6 = 34,012,224\) M-bytes. The computation of corresponding C-space obstacle is infeasible. Theoretically, various versions have been shown that the motion planning problem is \(\text{NP-complete} (\text{Latombe 1991})\). This means that the solution for the worst case will be exponential in computing time, and such a solution will be impractical in real applications for a large number of degrees of freedom.

1.3 Motivation

The difficulties of the motion planning problem have limited the development of motion planners for manipulators with a large number of degrees of freedom. Most proposed motion planners are not directly applied to manipulators with large numbers of degrees of freedom. These planners are often with manipulators with less than 6 degrees of freedom, or using some special treatment for several degrees of freedom, or using stochastic techniques. Chapter 2 will present a brief summary and analysis on the drawbacks of some proposed motion planners.

In practice, the real interest is to develop practical motion planners for manipulators with large numbers of DOFs. By practical motion planners, we mean that a planner will satisfy the following requirements:

- It requires moderate memory to finish planning tasks. For example, a workstation equipped with few tens of M-Bytes of memory, such as SUN SPARC station, SGI Indigo etc.

- It finishes planning tasks within a moderate time on existing workstation-type machines. For example, in the order of minutes on a SGI Indigo station.

- It can find a path with high probability if a path exists.
Our main motivation in this thesis is to develop a motion planner that satisfies the above practical requirements. A practical goal is that the planner can accomplish path planning for a 7-DOFs manipulator SPDM\(^2\) (Special Purpose Dextrous Manipulator) – which is proposed for the Space Station in the future.

### 1.4 An Overview of the Sequential Framework

The sequential framework (Gupta 1990; Gupta and Guo 1992) provides a general deterministic paradigm for developing practical motion planners for manipulators with a large number of degrees of freedom. In 1992, Ching and Badler applied this approach for a human-like robot (Ching and Badler 1992).

The essence of the sequential search framework is to exploit the serial nature of the successive structure of manipulator arms. Most of the manipulators in use are serial in structure. Serial manipulators have an inherent ordering of links, from the base link to the last link. The strategy of the sequential framework is to plan the motion of a group of links successively, starting from the base link. Each of the motion planning stages for a group of links is based on planning the previous group of links. Thus, this strategy decomposes the \(n\)-dimensional problem of planning collision-free motions for an \(n\)-link manipulator into a sequence of \(s\) smaller \((m_j + 1)\)-dimensional sub-problems, where \(m_1 + m_2 + \ldots m_s = n\). The \(j\)th sub-problem corresponds to planning the motion of a sub-group of \(m_j\) links. Each sub-problem is dependent on the previous sub-problems, and backtracking is used in case no solution exists for a given sub-problem. Furthermore, each sub-problem is represented as a path search in a \(t \times C^{m_j}\) sub-space, where \(t\) is the parameter along the path and \(C^{m_j}\) denotes the \(m_j\) dimensional configuration space of \(m_j\) links in a group.

An overview of the sequential framework is given in Figure 1.1, where \(m_j = 1, j = 1, 2, \ldots, n\), i.e. the implemented planner plans for a single link in each sub-problem. Suppose that the motions of links 1 through \(i - 1\) have been planned, i.e., the path of the proximal end of link \(i\) is determined. We parameterize this path with a parameter

\(^2\)The SPDM model used in this thesis is developed by International Submarine Engineering Ltd., which is called SPDM Testbed Manipulator (STM).
CHAPTER 1. INTRODUCTION

The motion of link $i$ is now planned along this path by controlling the degree of freedom associated with link $i$ — a single-link problem which can be solved as a 2-dimensional motion planning problem in $t_i \times q_i$ space (i.e., $t \times C^1$ space, see Chapter 2). Physical obstacles give rise to forbidden regions in the $t_i \times q_i$ space and a path avoiding these regions corresponds to a collision-free path for link $i$.

This framework essentially solves the motion planning problem by combining a problem-decomposing strategy and a backtracking mechanism. The problem-decomposing strategy avoids the difficulty of representing $n$-dimensional $C^1$ space (such a representation is impossible for manipulators with large numbers of degrees of freedom). The backtracking mechanism is used to improve the completeness.

Section 4.3 gives the detailed discussion of an empirical complexity analysis for the implemented planner.

1.5 Previous Implementation

The sub-problem in each $t \times C^m_j$ sub-space plays an important role in the sequential framework. Gupta and Guo (Gupta and Guo 1992) presented a visibility graph (V-Graph) based representation of the $t_i \times q_i$ spaces and a backtracking mechanism based on edge deletion within the sequential search framework (for the case of $m_j = 1, j = 1, 2, ..., n$). In this implementation, a CAD type geometric polyhedral representation is used for describing the workspace obstacles and the manipulator. The fundamental computation in determining the forbidden regions in $t_i \times q_i$ spaces is similar to (Lozano-Pérez 1987), i.e., for each discrete value of the parameter $t_i$, the forbidden intervals of $q_i$ are analytically determined. The forbidden regions are then approximated by polygons. A V-Graph is then constructed and searched for a collision-free path for link $i$. The main drawbacks of this representation are (i) the approximation of forbidden regions by polygons may result in a rather large number of nodes in the V-Graph (in the order of 200 to 300 nodes) in each $t_i \times q_i$ space, and (ii) a collision-free path invariably passes too close to the obstacles, an intrinsic property of the V-Graph representation. Furthermore, the backtracking mechanism is based on an edge deletion
CHAPTER 1. INTRODUCTION

Figure 1.1: An overview of the sequential framework, where each link is a sub-group. Max Level is the maximum level of backtracking and is a user determined parameter. B Level is the current level of backtracking.
mechanism in the V-Graph, and it is not very efficient because the size of the V-Graph is normally very large. If the V-Graph is sparse, an ad-hoc mechanism based on random node addition is used.

1.6 A Novel Approach within a Sequential Framework

This thesis presents a new approach within the sequential framework. In this new approach, a discrete bitmap representation is adopted for the subspaces (inspired by (Latombe 1991)). Then a numerical potential field defined over the bitmap is used to guide the path search for the sub-problem. The potential field is derived from the Voronoi diagram skeleton of the free space of the subspace, which guides the path search preferring the valley of the free space and leading to a safe path for the planning links. Such a bitmap-based representation and search have lead to several advantages for the sequential framework. Pitfall (i) in the previous V-Graph based implementation is avoided since no polygonal approximation is used, and pitfall (ii) is avoided since the paths follow the valleys of the the numerical potential fields and hence lie in the middle of the free-space, i.e., they tend to be safe paths (Latombe 1991).

Furthermore, this thesis introduces a novel backtracking mechanism that is based on inserting “virtual forbidden regions” in these sub-spaces. A virtual forbidden region inserted in the sub-space actually divides the original (topologically) equivalent class of collision-free paths into two. Thus the new backtracking mechanism efficiently explores the C-space for new paths in a more uniform and systematic manner and does not need any ad-hoc mechanism even if a sub-space has few obstacles.

Various motion planners (with $m_1 = m_2 = \ldots = m_n = 1$) have been implemented, and have been tested with hundreds of experiments for different kinds of manipulator arms including the 7-DOF SPDM arm. The motion planners are written in C and their average run time is within few minutes on a Sparc 10 station and SGI Indigo II.

The empirical complexity of the motion planner is given in the following equation,
CHAPTER 1. INTRODUCTION

\[ O(C^n p^b f) \]  

(1.1)

where \( n \) is the number of degrees of freedom, \( C \) is a constant (experimentally, it averages between 1.0 and 2.5), \( p \) is the maximum number of backtrackings in the sub-problems for each link, and \( b \) is the maximum level of the planner's backtracking to previous link's problem. \( f \) is a complexity of a single link problem.

We present a series of extensive experimental results for the motion planner based on the new approach for manipulator arms with up to eight degrees of freedom. For example, the planned motion for a 7-DOF STM (SPDM Testbed manipulator) arm is shown in Figure 1.2. Another example is the planned motion for a planar 8-DOF arm shown in Figure 1.3. The experimental results show that this new bitmap-based approach with the novel backtracking mechanism is faster and more robust than the previous V-Graph based implementation of the sequential framework.

In summary, the new approach within the sequential framework possesses several advantages: (i) It avoids the difficulties of representing n-dimensional C-space obstacles and of searching in the n-dimensional space. (ii) Its complexity, although exponential in \( n \), involves only a small constant (around 1.5). (iii) It is efficient and practical because the systematic search allows the planner to succeed with very limited number of backtrackings at very few levels. (iv) It is deterministic (i.e., no stochastic techniques are involved). (v) It plans safe paths for manipulator arms.

The main contributions of this thesis can be summarized as followings:

- Implementation of a new approach within the sequential framework.

- An empirical complexity analysis of the sequential framework reveals the real complexity from the polynomial appearance of previous analysis for the V-Graph-based implementation.

- Extensive experiments on the implemented planners with various manipulators with up to 8 degrees of freedom.
Figure 1.2: An example of planned motion sequences for a 7-DOF STM arm.
Figure 1.3: An example of planned motion sequences for an 8-DOF manipulator in a 2-D workspace.

The extensive experimental results empirically demonstrate that the new approach, developed within the sequential framework, is practical and efficient for manipulators with fairly large numbers of degrees of freedom.

1.7 Organization of the Thesis

This thesis is organized as follows. Chapter 2 reviews algorithms for robot motion planning presented in recent years, and briefly summarizes several representative algorithms. Chapter 3 concentrates on the solving of single link-problem based on the bitmap-based method. Chapter 4 demonstrates a backtracking mechanism for the novel bitmap-based approach and its implementation, and provides an empirical analysis of its complexity. Chapter 5 provides the experimental results and analysis. Finally, chapter 6 summarizes the new approach and provides suggestions for future research.
Chapter 2

Related Work

Motion planning has received widespread attention in the past fifteen years or so, and various algorithms have been presented for different situations. Latombe (1991), Hwang and Ahuja (1992) presented a detailed and exhaustive survey in this area. Because dealing with the high-dimensional complexity of the motion planning problem is very difficult, very few practical motion planners have been developed for manipulators with many degrees of freedom. Many of the implemented motion planners (Lozano-Pérez 1987; Lozano-Pérez and O’Donnell 1991; Hasegawa and Terasaki 1988; Kondo 1991) were for manipulators with less than six degrees of freedom and are not directly extended to more DOFs cases. Most recently, SANDROS planner (Chen and Hwang 1992) has been shown to be successful in reducing the computing time and memory requirement by using a two-level subgoal network search scheme. On the other hand, efforts have been concentrated on stochastic and parallel processing techniques (Barraquand and Latombe 1991; Latombe 1991; Lozano-Pérez and O’Donnell 1991; Challou, Gini, and Kumar 1993; Ahuactzin and et al. 1992). Results of some proposed approaches are very promising, however there is a long way from providing a satisfactory solution for the motion planning problem.

This chapter will first give a basic overview of the theoretical aspects of the motion planning problem, and then present a brief survey of recent research in motion planning algorithms and their implementations. Because of the mixed usage of various strategies in the proposed algorithms, it is difficult to classify them distinctly.
The categories in this overview are intended to help us to easily understand the main ideas in the algorithms. In addition, this survey concentrates on the motion planning approaches for manipulator arms with large numbers of DOFs.

## 2.1 Theoretical Aspects

Theoretical aspects of the motion planning problem have been extensively studied (Latombe 1991; Hwang and Ahuja 1992). The early representative research in this field can be traced to Schwartz and Sharir's contribution in 1983 (Schwartz and Sharir 1983b). They developed an algorithm for the generalized mover's problem. Their algorithm has a double exponential time complexity with the number of DOFs. (Canny 1987; Canny 1988) improved the complexity with a single exponential time. Canny showed that the motion planning problem for a manipulator arm with $n$ degrees of freedom under $m$ geometric features (the number of faces, edges etc. that describe the obstacles and manipulator arms) could be solved in time $O(m^n \log m)$.

This exponential complexity (in the degrees of freedom of the robot) has been accepted to be the upper bound of the motion planning problem in this field. The base $m$ is quite large (usually in the order of hundreds, even thousands). If the number of degrees of freedom for a manipulator arm is fairly large, (for example, some proposed manipulators used in space have more than 10 DOFs), the exponential increase becomes impractical for most manipulators. Thus the motion planning problem becomes computationally infeasible for a complete solution for large numbers of DOFs.

In spite of the intractability for a complete algorithm, researchers have developed different practical motion planners by using various heuristic and computational techniques in past twenty years or so. C-space with irregularly bounded C-space obstacles is extremely difficult to represent and search efficiently in a high dimensional case. In summary, efficient representations and search strategies that avoid the full exploration of the C-space are the key to developing practical motion planning algorithms.

Following sections briefly summarize some of practical motion planners by categorizing them by the main computing technique adopted. Our categories are certainly not complete, our purpose is to draw a brief picture of the state of the art motion
2.2 Practical Motion Planners

Motion planners for manipulator arms were developed as early as the late seventies, such as Lozano-Pérez's fundamental idea about configuration space (Lozano-Pérez and Wesley 1979), Brooks' cone-decomposition of free space (Brooks 1983), Donald's method of hypothesizing channels through free space (Donald 1983; Donald 1984), and Herman's swept-volume approach (Herman 1986). Some of the more recent ones are Lozano-Pérez's C-Space Approach (Lozano-Pérez 1987; Lozano-Pérez and O'Donnell 1991), Faverjon and Tournassoud's Octree Approach (Faverjon 1984; Faverjon and Tournassoud 1987), Barraquand and Latombe's Potential Field Approach (Barraquand and Latombe 1990; Barraquand and Latombe 1991; Latombe 1991), Kondo's Enumeration Approach (Kondo 1991), and Chen and Hwang's two-level subgoal network planner. Attempting to overcome the exponential time complexity of the motion planning problem, a number of researchers have implemented motion planners which apply parallel processing techniques (Lozano-Pérez and O'Donnell 1991; Challou, Gini, and Kumar 1993; Ahuactzin and et al. 1992). Two main aspects distinguish the existing motion planners from each other: (i) Building and representing approaches for the configuration space, and (ii) Searching methods for finding a collision-free path based on the representing approaches. Existing motion planners can be roughly classified into four categories (overlapping) by their emphasis on different representations and search mechanisms of the configuration space: (i) explicit computation of configuration space based planners (including our sequential framework), (ii) potential field based planners, (iii) multi-level representing and searching based planners, (iv) parallel computation based planners.
2.3 Explicit Computation of Configuration Space Based Planners

The concept of the configuration space is fundamental to the motion planning problem, since it is the space of all possible motions of a manipulator. Lozano-Pérez first presented the configuration concept for the spatial planning problem (Lozano-Pérez 1981; Lozano-Pérez 1983). Thereafter, the configuration space concept has been widely used in the motion planning area. The C-space concept shows that planning a collision-free path for an n-DOF robot is equivalent to finding a continuous path within a free space in an n-dimensional space defined in terms of the degrees of freedom. Figure 2.1 gives an example of a simple 2-DOF planar arm working in a 5-obstacle workspace. The 5 simple square obstacles give rise to quite irregular shaped C-space obstacles. For larger numbers of DOFs, the shape of C-space obstacles is very complicated.

Figure 2.1: C-space obstacles and path for a simple 2-DOF manipulator. (a) A manipulator with two revolute joints and its 5-obstacle workspace. (b) The corresponding C-space obstacles (the black regions) and C-space path. The 6 numbers in both figures correspond to 6 configurations in both spaces.
Many planners have been developed using the C-space based approach (Lozano-Pérez and Wesley 1979; Lozano-Pérez 1983; Lozano-Pérez 1987; Faverjon 1984; Faverjon and Tournassoud 1987; Kondo 1991). Our sequential framework is also in this category. Following section will briefly review few representative planners based on C-space approach.

2.3.1 Lozano-Pérez’s C-Space Approach

Lozano-Pérez (1987) applied the configuration space approach for manipulator arms. For an \( n \)-dimensional case, the C-space is represented by a union of \((n-1)\)-dimensional slice projections. A slice projection of a one-dimensional C-space region is defined by a range of values for one of the defining parameters of the C-space (the joint angles for a manipulator arm with rotary joints). In the implementation, the free C-space is represented by a multi-branch tree. The C-space obstacles are computed link by link starting from the first link considering the previous link’s discretization. Carrying on, a multi-branch free-space tree is obtained. Further, the free C-space is grouped into maximal regions, each region containing a kernel. A kernel is defined as part of the region that can be reached from any point in the region by moving in a straight line to one of the axes in the C-space. A collision-free path between two points inside same region is simply going straight into the kernel, moving along the kernel, then going to the goal. The connectivity among all regions is computed in a region graph. Finally, the well-known A* search (Hart, Nilson, and Raphael 1968) is used to find a solution. A planner has been implemented for this approach. It solved problems for a PUMA manipulator arm with six degrees of freedom, but the last three links and end effector were replaced by a bounding box. It is actually a problem with only four degrees of freedom.

The key step in Lozano-Pérez’s approach is to compute one-dimensional slice projections of the free C-space, that is, to determine the free ranges of one joint parameter for given values for all previous joint parameters. Theoretically, any motion planning problem with \( n \) degrees of freedom can be solved. However, such an exhaustive computation of the C-space is infeasible either in computation time or in memory requirement for manipulator arms with more than four degrees of freedom. For such
cases, Lozano-Pérez suggested that the last few links and the end-effector could be replaced by a simple bounding box. However, for cluttered cases or irregular load shapes, the bounding box will not be a good approximation, and the approximation will lose quite a large solution space and lead to failure. In addition, this approach is not suitable for manipulators whose distal links are larger in size than the proximal ones. An example of such a manipulator is the SPDM arm used in this thesis.

2.3.2 Octree Representation Approaches

Faverjon (1984) presented a hierarchical octree representation for the C-space. An octree is a tree of degree eight which hierarchically describes the space contained in a cube that forms the root. The children of a node are the eight cubes obtained by partitioning the parent node. The nodes are labeled as Full, Empty, or Mixed depending on whether they lie entirely in an obstacle entirely, outside of all obstacles, or partly in an obstacle. Only the Mixed nodes will be explored again until the final resolution size for the cubes is reached. Using collision-detection algorithms, physical obstacles in the workspace are transformed into its configuration space. Faverjon implemented this approach for the first three links of a manipulator arm. So the configuration space explored is 3-dimensional. The whole 3-dimensional C-space is divided into 64 x 64 x 64, and then filled up using the transforming algorithm. The scheme of searching a short collision-free path is based on the A* algorithm. Unfortunately, facing similar problems to the Lozano-Pérez’s approach, this approach can not be directly extended to planning a collision-free path for manipulator arms with large numbers of degrees of freedom, since both the memory requirements and the search time grow exponentially with the degrees of freedom. They suggested an extension to higher DOFs (see section 2.5.1).

2.3.3 Enumeration Approach

Kondo (1991) suggested an enumeration approach to the motion planning problem. The idea of the enumeration approach is to restrict the exploration of the configuration space to only cells that need to be explored for a free path during the search, thus
reducing the number of unnecessary collision detections. In the approach, initially
the C-space is equally quantized into cells by placing a regular grid, and the unknown
cells are explored hierarchically only when needed. A set of parameterized heuristic
functions is defined so that the A* search moves the manipulator from the start to the
goal, and the manipulator has a preferential moving direction which depends on the
heuristic parameters. The parameters are dynamically changed during the searching
according to the progress the manipulator has made towards the goal. A bidirectional
search is also used. Initially, all cells are unknown except for the free-space cells
containing the start and goal configurations. The free-space cells connecting the
initial and final configurations are enumerated based on the heuristic graph search
algorithm which is similar to the A* algorithm. A collision detection procedure is
called every time when the search algorithms explore the configuration cell. Mixed
(other than full-free or full-obstacle) cells are hierarchically divided.

This algorithm solved motion planning examples for a manipulator with 6 degrees
of freedom in a 3-dimensional workspace in average of 10 minutes on a SUN station.

The approach is fast when only a small portion of the free C-space cells are enu-
merated during the search process. However, in the more cluttered workspace, more
backtracking motion is certainly needed, and this approach will not be as efficient.
Furthermore, for an n-DOF manipulator arm, even a portion of discrete C-space
cells that are enumerated during the search could require a huge amount of memory.
Hence this approach is not extensible to manipulators with large numbers of degrees
of freedom.

2.4 Potential Field Based Motion Planners

Khatib used an artificial potential field to avoid imminent collisions among manipula-
tors and obstacles (Khatib 1985). The algorithm is mainly for the local avoidance of
obstacles in real time rather than for planning global paths. The artificial potential
field is defined over obstacles exerts a repulsive force when any link of the manipulator
approaches the obstacle. The goal position acts as an attractive force on the manipu-
lator. In this field of artificial forces, the manipulator end-effector (treated as a point)
moves to its goal position. The core of this approach is to set up artificial potential fields over the robot workspace. Each of these fields applies to a specific point on the robot. The robot then moves along the gradient direction generated by the artificial potential fields. The approach can actually be used to refine a given global path, as the potential field is to "bend" a planned global path. Unfortunately, this approach is not suitable for global path planning because of the local nature of the way the potential field is defined. The search guided by the potential field unavoidably becomes trapped in local minima very frequently. On the other hand, escaping from these local minima is not trivial.

Due to the complexity of the $n$-dimensional configuration space, a good potential field over the C-space is impossible to define for high dimensional cases. Most of the potential field algorithms are local-based.

2.4.1 Randomized Path Planner

Barraquand and Latombe (Barraquand and Latombe 1990; Barraquand and Latombe 1991; Latombe 1991) proposed a randomized approach based on numerical potential fields. Their algorithm incrementally builds a graph connecting the local minima of the potential field defined over the C-space, and concurrently searches this graph for the goal. A random motion is used to escape these local minima. The C-space potential field is a weighted combination of a series of potential fields defined in the workspace. Each of the workspace potential fields apply on a selected point on the manipulator, called a control point. The algorithm has been successfully applied to several manipulators with large numbers of degrees of freedom. This implemented planner is called Random Path Planner (RPP).

The stochastic nature of the random search has its intrinsic limitations. For example, the execution times may vary drastically even for the same example on different runs of the planner. In addition, as Barraquand and Latombe pointed out, "The efficiency of the RPP results from the fact that a typical path planning problem has many solutions, so that a globally random search procedure can find one if it (the C-space
potential field) is well informed." However, due to the complex high-dimensional C-space, a "well informed" potential field in the C-space can not be always successfully formed. In such situations, the C-space potential does not "aid" the search toward the goal, this approach will purely be a random search. Figure 2.2 shows a potential field that is almost useless for a 2-link arm. The initial configuration corresponds to (45, 110) (right side in Figure 2.2a) and the final configuration corresponds to (-210, -118) (left side in Figure 2.2a). The C-space potential field is generated by a control point on the tip of the arm. Note the potential field in the C-space (Figure 2.2b) gradually increases until a close neighborhood of the goal configuration is reached. Clearly the potential field is of no help in guiding the search to the goal except in the vicinity of the goal, a clear example that the potential function is not well informed. For large numbers of DOFs, the C-space potential function is not likely to be "well informed" in large scale. Zhu and Gupta (1993) presented some experiments that highlight the limitations of this approach.

2.5 Multi-Level Based Planners

2.5.1 Global-local Planner

Faverjon and Tounassoud (1987) proposed a two-level global-local planner. The global planner is essentially a very coarse discretized representation of the full C-space. Transitions between the adjacent cells are weighted (and updated) by the probability for the local planner to succeed in moving the system from one cell to another. These probabilities are used by the global planner (a minimum cost finding algorithm) to generate subgoals for the local planner. The local planner is based on a variant of the potential field technique. They have reported some good results for a full six degrees of freedom arm carrying a bulky load, and for a 10-DOF manipulator among vertical obstacles. Although it is possible to apply this technique to manipulators with many degrees of freedom, the global planner resolution would be very coarse for a given memory size and the probabilities may not converge. For example, they use a resolution of 16 degrees for a six degree of freedom problem. In fact, for much larger numbers of degrees of freedom problems, or in a cluttered workspace, the global
Figure 2.2: A C-space potential field that will not aid the random search. (a) 2-link arm in its initial and final position. (b) C-space potential.
CHAPTER 2. RELATED WORK

planner based on the coarse discretization will be virtually useless.

2.5.2 SANDROS Planner

Chen and Hwang (1992) proposed a two-level subgoal network motion planner, called SANDROS, which is an abbreviation for “Selective And Non-uniformly Delayed Refinement Of Subgoals”. Planning in two levels, the global planner keeps information about reachable, unreachable, and potentially reachable subgoals (portions of the C-space), and the local planner is used to determine the reachability of subgoals from a configuration point. During the planning, the global planner first generates a plausible sequence of subgoals to guide the search, and the local planner tests the reachability of these subgoals. If the local planner reaches every subgoal in the sequence, then a path is found. Otherwise, the global planner first tries to generate another sequence. If no sequence is available, the current subgoals are refined repeatedly until either a sequence is generated or the subgoals are refined to the resolution. A bidirectional best-first search strategy is adopted from both the start and goal positions.

This two-level hierarchical planning scheme has successfully reduced the memory requirement, which is normally a big barrier with large numbers of DOFs. Comparing with the two-level planner in (Faverjon and Tournassoud 1987), Chen and Hwang’s approach has two major improvements: (i) The subgoals are not simply fixed coarse discretizations of the C-space, instead, they could represent a quite large portion of the free C-space and can be refined when needed. (ii) the subgoals are computed as needed, and the detailed path among reachable subgoals is not stored (and can be obtained by the local planner later). These improvements make the searching faster and memory requirement smaller. The major advantage of the SANDROS is that its performance is commensurate with the task difficulty because the hierarchical search stops at a very coarse level of the subgoal sequence for a simple task. Thus, it runs fairly fast for reasonably cluttered situations. The reported run time for 5 and 6-DOF manipulators is approximately 10 minutes on a 16 MIPS SGI workstation.

The SANDROS planner appears to be quite efficient considering their test results for a 5 and 6-DOF manipulator. However, the efficiency of the SANDROS planner
for more cluttered environments and larger numbers of DOFs is not known. The computational complexity will still increase exponentially with the number of DOFs.

2.6 Parallel Implementations

To overcome the time consuming computation in construction and searching of the configuration space for the motion planning problem, some effort has been put into using parallel processing techniques.

2.6.1 Parallel Computing of C-space

A notable early attempt in this field is the parallel computing of C-space (Lozano-Pérez and O'Donnell 1991). Lozano-Pérez and O'Donnell apply a connection machine to compute the C-space maps. Basically, the implemented planner is a parallel version of Lozano-Pérez's early approach (Lozano-Pérez 1987). The benefit of the parallel application certainly is the reduction of the construction time for the C-space. It took approximately only two seconds to compute $64 \times 64 \times 64$ C-space on a connection machine with 64k processors. However, simply applying the parallel machine to the problem does not solve the memory requirement difficulty. The C-space exploration is still limited in first three dimensions for a 6-DOF PUMA arm.

2.6.2 Genetic Approaches

Genetic algorithms have become popular along with the development of massively parallel architectures in recent years. Genetic algorithms are stochastic search methods which are inspired by the evolving process of natural systems. More recently, some genetic optimization based approaches have been used to develop planners with few degrees of freedom. A planner for a 2-DOF planar arm is presented in (Ahuactzin and et al. 1992), and a planner for a 3-DOF mobile planner is described in (Mazer and et al. 1992). The essential idea of the genetic algorithms is to express a path planning problem as an optimization problem and to use parallel processing techniques to
overcome the massive computations.

During the planning, landmarks are placed over the configuration space. Each time the planner places a new landmark as far as possible from landmarks previously placed. Connections among landmarks are recorded. The planner will fill the free configuration space connected with the robot’s initial position with landmarks and a solution will be found as the landmarks get closer and closer to the goal.

The principal appeal of the algorithm lies in its massively parallel implementation. There is no successful implementation for robot with large degree-of-freedoms yet and on a serial machine it is still unclear whether it is practical to deal with large DOFs. The problem is that placing landmarks over the free configuration space could still be a heavy burden for large DOFs, as the requirement for memory storage could be fairly large if many landmarks are needed.
Chapter 3

Solving the Single Link Problem

The sequential framework solves the motion planning problem using two main techniques: (i) decomposing the $n$-dimensional problem into a series of small dimensional subproblems and solving the subproblems; (ii) backtracking if the planner fails in a subproblem solving stage. Thus, two main issues are crucial to the efficiency and practicality of the implemented planner: (i) efficient representation and search for the $(m_j + 1)$-dimensional subproblems, and (ii) systematic and efficient backtracking mechanism among the decomposed sub-problems to improve the planner’s completeness. In this thesis, each of the decomposed sub-problems are formed with $m_j = 1$, i.e. solving a single link problem each time.

The first section of the chapter will provide a general terminology and notations for the description of the new approach and implementation. Section 3.2 reviews robot coordinate systems and forward kinematics of the manipulator links and obstacles. Following sections will present the new approach and detailed implementation of the single link problem.

3.1 Terminology and Notations

A manipulator is denoted by $A$ and its workspace by $W$. $A$ has an inherent ordering of links, from the base to the last link. The links of the manipulator are labeled in
increasing order from the base, that is, the base link (which is fixed) is labeled as \( l_0 \) or base; the subsequent links are numbered \( 1, 2, \ldots, n \) with the last link labeled as \( l_n \). The \( i \)-th joint parameter is denoted by \( q_i \). The full configuration of the manipulator is denoted by \( q \). The start and goal configurations are given as \( q^s \) and \( q^g \). \( q_i \) denotes the \( i \)-dimensional joint space vector \((q_1, q_2, \ldots, q_i)\). \( q_i^s \) and \( q_i^g \) denote the start and goal configurations of the first \( i \) joints. We represent the \( i \)-dimensional joint space trajectory as a vector valued function \( q_i(t) = (q_1(t), q_2(t), \ldots, q_i(t)) : [0, 1] \rightarrow [q_i^s, q_i^g] \), where \( t \) is a parameter. Further, we distinguish between the mapping \( q_i(t) \) and its graph \( Q_i \), where \( Q_i = (q_i(t), t) \). Similarly, \( Q_i = (q_i(t), t) \).

Two important technical points need be mentioned here. The first is that after having planned the motion of link \( i \), the entire path \( Q_i \) is reparameterized with a new parameter \( t_{i+1} \). This is needed since the path for link \( i \), \( Q_i \) may not be (and often will not be) a single valued function of \( t_i \). Therefore, \( Q_i \) is parameterized (in the process, re-parameterizing \( Q_{i-1} \) with a new parameter \( t_{i+1} \), resulting in a new parameterization of \( q_i(t_{i+1}) \). The final trajectory will be \( q_i(t_{n+1}) \).

The second point is that a link may need to move outside its specified start to goal interval \([q_i^s, q_i^g]\). In fact, in many situations, the only path may be for the manipulator to move back away (or forth forward) from the start (or the goal) position and then come back to it. To incorporate such backup movements within the sequential search, the path \( Q_i \) in \( t_i \times q_i \) space is augmented in the interval \([0, t_i^s]\) (called backward extension path \( Q_i^{brd} \)) and in the interval \([t_i^g, 1]\) (the forward extension path \( Q_i^{frd} \)), where \( t_i^s \) and \( t_i^g \) correspond to the start and goal values respectively of the parameter \( t_i \). A more detailed discussion can be found in (Gupta 1990).

### 3.2 Kinematics

In order to describe the position and orientation of an object (a manipulator link or an obstacle), a coordinate system (frame) is attached rigidly to the object. The position and orientation of this frame is described with respect to some reference coordinate system, each frame is with reference to the previous link's frame. Once the frames are determined for the robot links, the position and orientation of the link's frame can be
obtained by forward kinematics based on homogeneous transformation matrices.

Appendix A gives a brief description of the link coordinate systems, link parameters, and forward kinematics. In later algorithmic descriptions of the new approach and implementations, the implicit computation of the forward kinematics is omitted for simplicity.

### 3.3 A New Bitmap-Based Approach

For efficient and practical motion planning, this chapter proposes a new bitmap-based approach within the sequential search framework for the single link problem. Three main ideas are incorporated in the new approach: (i) A discrete bitmap-based representation for the $t \times q$ spaces. (ii) A safest path search guided by a potential field which is derived from the Voronoi diagram\(^1\) skeleton of the free space of the $t \times q$ space. (iii) An efficient backtracking mechanism among the single-link problems, which performs a binary-search-like search for the whole C-space. The third issue will be covered in the next chapter.

The following sections of this chapter and next chapter will introduce the details of the new approach (with solving single link problem in each stage) and its implementations. While dealing with a single link problem at each stage, i.e., planning one DOF, each sub-space is also called a $t \times q$ space.

### 3.4 Building Forbidden Regions in the $t_i \times q_i$ Space: The $t_i \times q_i$ Bitmap

The fundamental stage in this implementation of the sequential framework is to plan the path for a particular link, say link $i$, given that the collision-free path $Q_{i-1}$ for the previous $i-1$ links has already been determined. The path $Q_{i-1}$ is parameterized with a parameter $t_i$. The obstacles and previous links give rise to constraint on the

---

\(^1\)The term used in this thesis is actually referring to the nearest-site Voronoi diagram (Aurenhammer 1991).
motion of link $i$. Such constraints are represented as forbidden regions in the $t_i \times q_i$ space. Planning a collision-free path for link $i$ is then to find a path, from $(t_i^0, q_i^0)$ to $(t_i^f, q_i^f)$, in the $t_i \times q_i$ space avoiding the forbidden regions in this space.

3.4.1 The $t_i \times q_i$ Bitmap

A discrete bitmap is used to represent the $t_i \times q_i$ space. The size of $q_i$ dimension is determined by a given resolution, e.g., with a resolution of 2 degree, the dimension, $\dim q_i$, of $q_i$ will be $360/2 = 180$ (assuming no joint limits). The size of $t_i$ dimension is equal to the length (in pixels) of the path $Q_{i-1}$ of previous $i - 1$ links.

If a $t_i \times q_i$ cell in the bitmap drops into a forbidden region, i.e. link $i$ collides with obstacles or previous links at the corresponding configuration, the corresponding cell is labeled as a forbidden cell, otherwise it is free. In practice, the bitmap is obtained cell by cell by checking the collision of the $i$th link with workspace obstacles and previous links. For each $t_i$ value, the position of the distal end of link $i - 1$ is already determined by the previously planned path, and is computed using forward kinematics. Link $i$ can move with respect to the distal end using the degree of freedom associated with it. The collision status of link $i$ with the obstacles and with previous links is checked for each $q_i$ value at the resolution. If no collision is detected for a $q_i$ value, the corresponding cell $(t_i, q_i)$ in the $t_i \times q_i$ map is assigned free. A typical $t \times q$ bitmap for a 4 DOF planar arm is shown in Figure 3.2. The corresponding initial and final configuration of the manipulator are shown in Figure 3.1. The dark areas are the forbidden regions. Two crosses in each $t \times q$ map represent the start $(t_i^0, q_i^0)$ and the goal $(t_i^f, q_i^f)$. Note that the start and goal in the right $t \times q$ space belong to different connecting sets.

3.4.2 Collision Detection in 2-Dimensional Workspace

Two kinds of collisions should be considered in the collision detection for link $i$: the collision of link $i$ with the obstacles; and the collision of link $i$ with previous links, i.e., self-collision.
CHAPTER 3. SOLVING THE SINGLE LINK PROBLEM

Figure 3.1: An example of a 4-dof arm in its 2-dimensional environment: (a) initial position and (b) final position.

Figure 3.2: Examples of two $t \times q$ forbidden region maps, where the bright area is free.
CHAPTER 3. SOLVING THE SINGLE LINK PROBLEM

Collisions with the Obstacles

For a 2-dimensional workspace, collision detections and distance computations are based on bitmap based representations of \( W \) (similar to (Latombe 1991)). Assuming that \( W \) is a bounded subset of \( \mathbb{R}^d \) \((d = 2 \text{ in this implementation})\), a 2-dimensional workspace \( W \) can be modeled as a 2-dimensional array at a given resolution. For example, a 2-D workspace can be discretized as a 256x256 bitmap at a 1/256-resolution of the workspace. The distance of the link \( i \) to obstacles is used to detect collisions. If this distance is within a safety distance from the obstacles, a collision is declared.

The 2-dimensional workspace implementation uses a recursive strategy to obtain a conservative distance for a link. First, a distance map over \( W \) bitmap is pre-computed to speed up the computations. This distance map is defined with values at each of the free cells that are the nearest distance from the cell to the obstacles. The distance of link \( i \) to the obstacle set is defined as the minimum distance of any point on link \( i \) to any point on the obstacle set. Collision checking between link \( i \) (at a given configuration \( q_i \)) and the obstacles can be done efficiently using a recursive dividing strategy.

Assume link \( i \) is a line segment with length \( L_i \). The distance values for the two end points of the link (in configuration \( q_i \)) are easily obtained by looking up the distance map. If both values are greater than the half length \( L_i/2 \) of the segment, it implies that link \( i \) (in configuration \( q_i \)) does not collide with the obstacles. If either of the two distances is less than or equal to zero (or, a safety threshold), link \( i \) is in collision with the obstacles. If none of the above cases applies, the segment is sub-divided into two halves, and the test is repeated for each half with the half length of \( L_i \). The process is carried on until the finest resolution is achieved.

For polygonal links, the collision between the link and the obstacles can be checked by repeating the above procedure for every edge of the polygon. This strategy is effective and efficient, as long as the obstacles are not small enough to be completely enclosed inside a link and the resolution for the link joint parameters is fine enough to avoid the link skipping over thin obstacles. Otherwise, the whole polygon needs to be sub-divided into smaller polygons. In this thesis, the 2-dimensional implementation is mainly to study the algorithmic issues of the new approach, so checking all the edges
is assumed to be sufficient.

In real implementation, the workspace distance map is computed using the $L^1$ metric. The $L^1$ metric is a conservative approximation of the real distance, and since we are interested in only whether a collision occurs or not (not the actual distance), such an approximation is more than adequate for this purpose. The main reason to use the $L^1$ metric instead of $L^2$ is that the $L^1$ distance is easier and faster to compute for the discrete bitmap case. The construction procedure of the $L^1$ distance map is similar to the one used in the $t_i \times q_i$ space in Section 3.5.

**Self-collision Check**

In addition to checking the collision of link $i$ with obstacles, the collision of link $i$ with the previous links should be checked for the self-collision. Self-collisions of link $i$ with links $1, 2, \ldots, i - 1$ are also mapped as forbidden regions in the $t_i \times q_i$ space. Using forward kinematics, all the previous links are mapped onto a bitmap of the same dimension as the workspace bitmap. For a given $q_i$ value, a collision-check is then performed for link $i$ by checking whether the mapping of the link $i$ is overlapped with the mapping of previous links on the self-collision bitmap.

In the real implementation, an 8-bit array is used as a collision check bitmap, each bit is used for one link, i.e., bit 0, bit 1, ..., and bit 7 correspond to link 0, link 1, ..., and link 7, respectively. For efficiency, instead of mapping the whole polygonal region on to the map, only the link’s edges are mapped onto the bitmap. This is based on the assumption that one link can not be completely enclosed in another link, and that the resolution for the links’ parameters is sufficiently fine. If the above assumptions are not true, mapping the entire link region should be used, and will result in somewhat slower self-collision checking. For the same reason as given for the collision detection of link with the obstacles, such situations are ignored.
3.4.3 Collision Detection in 3-Dimensional Workspace

For 3-dimensional workspaces, a straightforward extension of the collision detection method used in 2-dimensional implementation will have to work at least on the surface of polyhedra instead of line segments, and the recursive strategy will not be as efficient. This is because the distance computation for a surface region is more complicated than for a line segment in a distance field, which possibly leads to an increase of a magnitude for the very fundamental computation of the recursive strategy.

There are several proposed algorithms for collision detection and the closely related problem of computing distances between polyhedra. As we show later, distance information is not necessary but can be used to speed up the detection of collisions. The collision detection is also referred to the intersection problem in computational geometry. Some early works can be traced to 1978, an \(O(n \log(n))\) algorithm was proposed for finding the intersection of two convex polyhedra (Muller and Preparata 1978). Another \(O(n \log(n))\) algorithm using space sweeping strategy can be found in (Hertel and et al 1984). By using a hierarchical description of polyhedra (Dobkin and Kirkpatrick 1985), the complexity is reduced to linear. Cameron and Culley (Cameron and Culley 1986) presented a fast algorithm for computing the minimum distance between two polyhedra, which is believed to be suitable for real-time smooth animation of robot applications which contain dozens of polyhedra (Culley and Kempf 1986). A linear algorithm was developed for computing the distance between general polytype objects (Gilbert, Johnson, and Keerthi 1988). Recently, Lin and Canny presented an algorithm for incremental distance computation, and reported that the running time is roughly constant for finding the closest points between two polyhedra while the nearest points are approximately known (Lin and Canny 1991). Such cases are common in robot motion simulation and animation.

The 3-dimensional implementation in this thesis uses hierarchical representations

\[\text{\textsuperscript{2}}\text{The recursive dividing strategy could still work by sub-dividing a polyhedron into a set of smaller ones for detecting collisions with obstacles. However, it will not work for the self-collision checks. The reason is that for self-collision checks for link }i, \text{ we will likely have to compute the distance between the components of link }i \text{ and previous links of the arm. In order to perform the recursive distance look-up strategy, we have to keep all distance maps for previous links that need to be updated frequently as the previous links are moving too. This dynamic updating is a huge burden that makes self-collision checks impractical using such a recursive strategy.}\]
Figure 3.3: A hierarchical model for several obstacles, where a dotted-line parallelepiped object encloses three simple objects.

of the manipulator and the obstacles combined with efficient polyhedral intersection detections similar to (Faverjon and Tournassoud 1987) and (Gilbert, Johnson, and Keerthi 1988) for efficient computation of the forbidden regions.

Hierarchical Geometric Model

The 3-D implementation makes use of the programming and modeling facilities provided by ACT robot software. The ACT programming environment provides tools for hierarchical representations of the geometrical models of the workspace environment and manipulator arm, and for the efficient distance computation between polyhedra.

The hierarchical models are built to use simple parallelepiped objects to enclose a set of more detailed objects. Figure 3.3 shows a hierarchical model of a simple enclosure for three obstacles. These simple objects can be split recursively to obtain the simplest primitives at the leaves of the hierarchical tree. (See (Mazer and et al 1991)) The obstacles in the workspace and the links of the manipulator are unions of

\[ \text{ACT Software is a registered product for robot programming developed by ALEPH Technologies.} \]
simple primitives. A significant benefit of using the hierarchical representation model is that the hierarchical structure can avoid unnecessary detailed computation while the related objects are away from each other, i.e., the distance computation procedure will not look into the detailed level of the hierarchical representation and will keep the simple geometrical enclosure until the distance between the related objects is close enough to be of interest. This distance threshold is called the distance limit range, i.e., the hierarchical representation will keep the coarsest possible representation of the model until the distance between two coarse-represented primitives is shorter than the limit range.

For the motion planning problem, we are interested in only whether a collision occurs at certain configurations, not what the exact distance is from manipulator arm to the environment. A conservative distance approximation is more than adequate while the manipulator is relatively far away from obstacles. For the $t \times q$ bitmap construction in this 3-dimensional implementation, the distance computation range is limited to the radius of the link. For example, the link length could be simply chosen as the distance limit range if it is the dominant size for this link. Self-collision checking is done in the similar way by computing the distances between the current planning link and previous links.

The Speed-Up of $t \times q$ Space Computations

While computing the forbidden bitmap of the $t_i \times q_i$ space, a distance from link $i$ to the environment and previous links at a specific configuration gives more information than merely whether a collision occurs or not. Actually, the distance (even a conservative distance) is used to predict a free range for link $i$ rather than a single free cell. Figure 3.4 shows the computation for the free range. Suppose that $D$ is the distance of the link to the environment, and link radius is $L$. Since $D$ implies that the link is free to move within such a distance, at least a free range $q_{free} = D/L$ in radians from this position of link $i$ can be predicted by the distance $D$. Using such free ranges to determine a set of free cells for the $t_i \times q_i$ bitmap makes the $t_i \times q_i$ space computation

---

4The radius of a link is defined as the maximum distance from any point of the swept volume of the link generated by rotating the link around its joint axis to its joint axis.
Figure 3.4: Free range prediction. The distance and radius of a link is used to estimated a free range $D/L$ (Note the radius in the figure is an approximation).

much faster.

3.5 Numerical Potential Fields and Search in the $t_i \times q_i$ Space Bitmap

The next step in the algorithm is to define numerical potential fields over the bitmaps of $t \times q$ space. The potential fields are similar to (Latombe 1991) except that they are defined in $t \times q$ space, and not in the workspace. This section will briefly describe the computations here.

3.5.1 Build the $L^1$ Distance Map

First, an $L^1$ distance map is built for the $t_i \times q_i$ space bitmap by a wave-front expansion algorithm as follows. Suppose $C_{free}$ is the set of all free cells in the $t_i \times q_i$ space bitmap. The distance values of the boundary cells of the obstacles are initialized as zero. All the free neighboring cells of a cell (in $C_{free}$ ) with distance value $d$ are assigned $(d + \Delta d)$,
where $\Delta d$ is the $L^1$ distance between the neighbors. This is repeated until every cell in $C_{\text{free}}$ has been assigned a distance. A detailed description of the algorithm for the $L^1$ distance map can be found in (Latombe 1991). Examples of the $L^1$ distance maps for the $t \times q$ space are shown in Figure 3.5. A discretized Voronoi diagram (skeleton of the free space in the $t_i \times q_i$ space) of the $t \times q$ space is simultaneously extracted. The Voronoi diagram is a set of points that are equidistant from two or more points on the obstacles, which is a network of curves that lies “as far away as possible” from the obstacles. An example of Voronoi diagram for three obstacles within a square boundary is given in Figure 3.6. For the $t \times q$ space, due to the uneven boundaries of the forbidden regions, many small branches appear in the Voronoi diagrams (Figure 3.7), but these small branches do not affect the next stage, i.e., building the numerical potential fields over these bitmaps.

### 3.5.2 Numerical Potential Fields

A numerical potential field starting from $(t^0_i, q^0_i)$ is then derived from the Voronoi diagram in two wavefront procedures. The first procedure is to define a potential skeleton over the Voronoi diagram starting from the $(t^0_i, q^0_i)$. Let $V_0$ represent the set of the cells on the Voronoi diagram. In order to describe the algorithm, we first define the term neighbor in the $t_i \times q_i$ discrete grid.

In the general case of a $m$-dimensional grid, the $p$-neighbors ($1 \leq p \leq m$) of a configuration $q$ are defined as all these configurations in the grid having at most $p$ coordinates differing from those of $q$. The difference between two adjacent coordinate values is exactly one increment in absolute value. There are $2m$ 1-neighbors, $2m^2$ 2-neighbors, ..., and $3^m - 1$ $m$-neighbors. For our 2-dimensional $t_i \times q_i$ space grid, we can choose 2-neighbor as our definition of neighbor for a $(t^0_i, q^0_i)$ in the $t_i \times q_i$ grid.

An algorithmic description of the procedure is as follows:

1. Find a path $S$ from the point $(t^0_i, q^0_i)$ to the Voronoi diagram by following the gradient of the $L^1$ distance map. Let $V_i = S + V_0$ be the connected set containing the point $(t^0_i, q^0_i)$.
Figure 3.5: Two $t \times q$ distance maps. They correspond to the $t \times q$ spaces in Figure 3.2. Brighter area indicates larger distance values and the completely dark areas are forbidden.
CHAPTER 3. SOLVING THE SINGLE LINK PROBLEM

Figure 3.6: A Voronoi diagram for 3 objects in a square.

Figure 3.7: Examples of Voronoi diagrams in the \( t \times q \) spaces. They are generated for the \( t \times q \) spaces in Figure 3.2. Many of the small branches of diagrams are caused by uneven boundaries of the \( t \times q \) forbidden regions.
2. Label the point \((t^g, q^g)\) with zero.

3. Label the neighbors of the labeled cells in \(V_1\) with a value incremented by 1.

4. Repeat step 3 until all cells in \(V_1\) are labeled.

The second wavefront procedure tries to spread the potential field over all the free cells in the \(t_i \times q_i\) space bitmap. The procedure works as followings:

1. Let \(V_1\) be the labeled skeleton in the \(C_{free}\).

2. Label all neighbors in \(C_{free}\) of a labeled cell in \(V_1\) (say, with label \(l\)) that have not been labeled, with label \(l + 1\). Put the newly labeled cell in \(V_1\).

3. Repeat labeling until no more unlabeled neighbor cells can be found from \(V_1\) in \(C_{free}\).

After applying the above two procedures over the \(t_i \times q_i\) space bitmap, a potential field for the connected set of free cells containing \((t^g, q^g)\) in the \(t_i \times q_i\) space bitmap is built. Figure 3.8 shows two such potential fields that are generated for the \(t \times q\) spaces shown in Figure 3.2. Note the right potential field starting from the goal has just reached half of the free cells — the connected free space containing the goal.

### 3.5.3 Gradient Search

A gradient (Best-First) search is then used to find a path \(Q^\text{main}_i\) from the start point \((t^s, q^s)\) to the goal point \((t^g, q^g)\). Note that by construction there is no local minimum in the potential field for each \(t_i \times q_i\) space, thus it is guaranteed to find a path if there is one. Figure 3.9 shows a path found in the \(t_2 \times q_2\) space which keeps as far as possible from the forbidden regions. Although the search is a global computation, it is extremely fast (in a fraction of a second for a 256 \(\times\) 256 bitmap). If \(Q^\text{main}_i\) is found, then a forward extension \(Q^\text{frd}_i\) and a backward extension \(Q^\text{brd}_i\) are computed (see next section). Combining \(Q^\text{main}_i\), \(Q^\text{frd}_i\) and \(Q^\text{brd}_i\), a whole \(t \times q\) path \(Q_i\) is thus obtained. In conjunction with the path for previous links, \(Q_i\), (i.e., \(q_{i-1}(t_i), Q_i\)) completely
Figure 3.8: Two examples of the $t \times q$ potential fields. Brighter area represents larger potential values, the complete dark areas are the forbidden regions, and the half-dark areas are free but without potential.
determines $Q_i$, and is simply reparameterized by index of the $Q_i$ to be the parameter $t_{i+1}$ for the next link.

The numerical potential field based on the Voronoi skeleton has several advantages. Most importantly, the potential field spreads out over the free $t_i \times q_i$ space with priority of low potential values for choosing the valley of the free space. This property results in a trend for the later gradient search to avoid the $t_i \times q_i$ forbidden regions as far as possible, thus leaving possible maximum amount of space for the next link’s motions. Further, the potential field has no local minimum other than the goal $(t^g_i, q^g_i)$, and it guides the search to find a path if there is one. In addition, the potential field already provides the answer to the question of whether or not a path exists in the $t_i \times q_i$ space. If the potential has propagated to $(t^s_i, q^s_i)$ from $(t^d_i, q^d_i)$, a path exists. Otherwise, the $(t^s_i, q^s_i)$ and $(t^d_i, q^d_i)$ belong to different connecting sets in the $t \times q$ bitmap.

Figure 3.9: The bitmaps for a 4-DOF planar arm. The $t_2 \times q_2$ space and path are at left figure, the $t_3 \times q_3$ is at right. Note that there is no path for link 3.
3.5.4 Forward and Backward Extensions

A link may need to move outside its specified start and goal interval \([q^*_i, q^+_i]\). In fact, in many situations, a link has to do so in order to find paths for the later links. In such cases, links move to back away from the initial (or, the goal) position then come back to it. Such backup movements are called backward and forward extensions (Gupta 1990).

The determination for the forward and backward extensions is based on a simple method. The method is as follows: If \(q^*_i\) is less than or equal to \(q^+_i\) then the forward extension \(Q^\text{frd}_i\) is the path from \((t^*_i, q^*_i)\) to the closest point from the top-right corner \((1, q^+_i)\) which is reachable from the \(q^*_i\). This closest point is the very first point with an assigned potential value, encountered while searching in a breadth-first manner starting from the top-right corner. The backward extension is the path from \((t^*_i, q^*_i)\) to the closest reachable point from the bottom-left corner \((0, q^+_i)\).

If \(q^*_i\) is greater than \(q^+_i\) then \(Q^\text{frd}_i\) is the path from \((t^*_i, q^*_i)\) to the closest reachable point from the bottom-right corner \((1, q^*_i)\), and the \(Q^\text{brd}_i\) is \((t^*_i, q^*_i)\) to the closest reachable point from the top-left corner \((0, q^*_i)\).

Note the choices for the forward and backward extension are not unique. One can choose the opposite top/bottom corner for the extension end. The above choice is most likely to form a shorter path while maximally covering the whole range for \(q_i\) by the combined path.

To keep the consistency of the paths, these two extensions are obtained using the same technique as the main path \(Q^\text{main}_i\) in \(t_i \times q_i\) space. Cascading the backward extension \(Q^\text{frd}_i\), the main path \(Q^\text{main}_i\) and the forward extension \(Q^\text{frd}_i\), a whole \(t_i \times q_i\) path \(Q_i\) is obtained for the planning of next link.

3.5.5 Reparameterization of the \(t_i \times q_i\) path

Note the \(Q_i\) is not a single valued function of \(t_i\). To plan for the link \(i + 1\), \(Q_i\) is reparameterized with a new parameter \(t_{i+1}\). In our discrete representation, \(t_{i+1}\) is simply the index value of \(Q_i\) (see Appendix A).
CHAPTER 3. SOLVING THE SINGLE LINK PROBLEM

Normally, the length of \( Q_i \) will be larger than the length of \( t_i \) in pixels, thus causing a cumulative increase of the sizes of the \( t \times q \) spaces. In order to keep the given resolution the same for early degree of freedom, this reparameterizing strategy is necessary as the distance of any neighboring points on the path \( Q_{i-1} \) is in the finest resolution unit, and loss of a point on the path means planning for next link at a more coarse resolution. This gives rise to an exponential factor in the overall complexity of the algorithm, and will be discussed in section 4.3. In the previous V-Graph based implementation, a strategy called equal-length parameterization strategy (Guo 1992) was used to keep the lengths of all \( t_i \) the same for all \( t_i \times q_i \) space. However, subsequently we observed that some collision computation errors occurred in the obtained paths due to the resolution loss.

Once the single link algorithm determines whether a path has been found for the current link, the sequential framework will proceed upon the single link problem results. If a path for current link is found, the sequential framework plans for next link along the obtained path. If no path is found, then the sequential framework backtracks to previous link to try to find an alternative path. The backtrack mechanism is presented in the next chapter.
Chapter 4

Backtracking Strategy

While planning for the \( i \)th link, the sequential framework does not consider other distal links (link \( i + 1, i + 2, \ldots, n \)), therefore along a path found in a \( t_{i-1} \times q_{i-1} \) space, one may not find any path in the \( t_i \times q_i \) space. In such a situation, a backtracking strategy is applied. A main question here is how to generate a new path or modify the old path for link \( i - 1 \) if there is no path in a \( t_i \times q_i \) space. This chapter introduces a novel backtracking mechanism that is based on inserting "virtual forbidden regions" in \( t_i \times q_i \) spaces.

4.1 Determining Block and Virtual Forbidden Regions

Failure to find a path in a \( t_i \times q_i \) space implies that the start \((t_i^s, q_i^s)\) and the goal \((t_i^g, q_i^g)\) in \( t_i \times q_i \) space belong to different sets of the collision-free set of the \( t_i \times q_i \) space, i.e., the \( t_i \times q_i \) space is blocked between the start and goal. Figure 3.9 shows the \( t_2 \times q_2 \) and \( t_3 \times q_3 \) spaces for the example in Figure 3.1. A vertical forbidden region completely blocks any possible path from \((t_3^s, q_3^s)\) to \((t_3^g, q_3^g)\) in the \( t_3 \times q_3 \) space. Thus backtracking to link 2 is needed.

In order to modify the previous link’s path, we need to determine which section of the path of the previous link causes the block. This is readily detected during
the propagation of the numerical potential fields — the start position simply does not belong to the potential being propagated from the goal position shown in Figure 3.8(b), the potential field has not reached the start point in the \( t_3 \times q_3 \) space. Clearly, \( Q_{i-1} \), the original path in \( t_{i-1} \times q_{i-1} \) space needs to be modified. However, the geometry and topology of the blocks (in the \( t_i \times q_i \) space) can be quite diverse, as shown by some examples in Figure 4.1.

The way to generate a new path will eventually affect the backtracking efficiency. The strategy used in this new approach is to encode the entire geometry of the “block” into a single parameter value — \( t^\text{block}_i \). Note that \( t^\text{block}_i \) corresponds to a single point \( (t^v_{i-1}, q^v_{i-1}) \) on the path \( Q_{i-1} \) in \( t_{i-1} \times q_{i-1} \) space. A small forbidden region is then inserted at this point in the \( t_{i-1} \times q_{i-1} \) space. Therefore, all those paths in \( t_{i-1} \times q_{i-1} \) space that pass through (and in the neighborhood of) \( (t^v_{i-1}, q^v_{i-1}) \) are deemed infeasible. The \( t_{i-1} \times q_{i-1} \) space is searched again for a new path \( Q'_{i-1} \) for link \( i-1 \). Clearly, the new path \( Q'_{i-1} \) will avoid the inserted virtual forbidden region corresponding to the block in the \( t_i \times q_i \) space. In fact, since the search is based on potential fields derived from the Voronoi skeleton, \( Q'_{i-1} \) deviates as far as possible (given the constraints from other forbidden regions) from the old path \( Q_{i-1} \), at least in the vicinity of the virtual forbidden region inserted. Along this new path, \( Q'_{i-1} \) for link \( i-1 \), the \( t_i \times q_i \) space is built again for link \( i \), and then searched for a new path for link \( i \).

This scheme is illustrated with a simplified example. Suppose the free \( t_{i-1} \times q_{i-1} \) space is a channel, thus the path \( Q_{i-1} \) is a straight line segment in the center as shown in Figure 4.2. Suppose a virtual obstacle (represented by a small blob in the middle) is inserted at a certain value \( (t^v_{i-1}, q^v_{i-1}) \) during a backtracking. The new path would be one of the two new possible paths \( Q'_{i-1} \) shown in the Figure. The insertion of a virtual forbidden region, therefore, divides the original (topologically) equivalence class of collision-free paths into two. Repeated application of this backtracking mechanism, therefore, leads to a systematic search of the \( t_i \times q_i \) space.

It is possible that no path exists in the new \( t_i \times q_i \) space after one backtracking. In such cases, another virtual forbidden region will be inserted at the newly determined blocking position, and new alternative paths in the \( t_{i-1} \times q_{i-1} \) space are used again to try find a path in \( t_i \times q_i \) space. Figure 4.3 demonstrates four more alternative paths.
Figure 4.1: Examples of the “block”s in the $t \times q$ space.
Figure 4.2: Inserting virtual forbidden region.

For backtracking if the two paths \( Q'_{i-1} \) do not lead to a solution for link \( i \). Once a path is found in the \( t_i \times q_i \) space, the planner resumes the normal sequential planning for link \( i + 1 \).

This scheme implies a systematic exploration principle in the configuration space. If a collision-free path exists, there should be a volumetric free channel enclosing the single path curve in the \( n \)-dimensional C-space. Such a volumetric channel means that there is a feasible range for every link at a given free configuration. Normally, the more sparse the workspace is, the wider the feasible range. In the \( t \times q \) space, the feasible volume at a given configuration corresponds to a feasible region, and all paths in this feasible region will satisfy the need for the planning problem. The backtracking mechanism acts to modify an existing path to converge to a feasible region, and the modifications are done systematically. So the backtracking does not have to go into a more detailed level search of the \( t \times q \) space unnecessarily. Thus such a backtracking scheme saves unnecessary searches.

The forbidden region is virtual in the sense that it represents a constraint in \( t_{i-1} \times q_{i-1} \) space, not due to link \( i - 1 \) but due to link \( i \). However, note that it is an
approximation since it represents the fact that no configuration of link $i$ is collision-free at $(t_{i-1}^v, q_{i-1}^v)$. Geometrically speaking, it is equivalent to approximating the shape of the block in $t_i \times q_i$ space by a thin vertical strip at $t_i^{\text{lock}}$. For the example in Figure 3.9 of the previous chapter, Figure 4.4 shows that a virtual forbidden region (Figure 4.4.a) in the $t_{i-1} \times q_{i-1}$ space is equivalent to a forbidden strip in the $t_i \times q_i$ path built along the old path (Figure 4.4.b).

Because the shapes of blocks can be quite diverse, no one single choice of $t_i^{\text{lock}}$ will always be most effective. Therefore, a simple and fast strategy is used to determine a $t_i^{\text{lock}}$ as follows. The $t_i$ coordinate of the first forbidden cell (along the straight line segment, connecting the start and the goal point) which has at least one neighbor with potential value gives the blocked parameter value, $t_i^{\text{lock}}$. In practice, a cone-shaped region with cone angle $\alpha$ (Figure 4.5a) is swept along the straight line and the search is limited to a diamond shaped region starting from the start to the goal (Figure 4.5b).

An example is shown in Figure 4.6. Due to the block (right in the middle of $t_3 \times q_3$ space), the planner fails to find a path for link $3$ and determines that the block occurs
CHAPTER 4. BACKTRACKING STRATEGY

Figure 4.4: The inserted virtual forbidden region in the \( t_{i-1} \times q_{i-1} \) space is equivalent to a vertical block strip in the \( t_i \times q_i \) space.

at a certain value \( t_i^{\text{block}} \). A small virtual forbidden region (shown in black) is then inserted at the corresponding point in the \( t_2 \times q_2 \) space as shown at the bottom of Figure 4.6. Note that the small virtual forbidden region was absent in the \( t_2 \times q_2 \) space at the top of Figure 3.9. The new path in the \( t_2 \times q_2 \) space automatically chooses a wide channel from the start to the goal.

The planner may backtrack further if it exhausts all paths for link \( i - 1 \) as the cumulative effect of inserting virtual forbidden regions may completely block the \( t_{i-1} \times q_{i-1} \) space, or a maximum backtracking number is reached in this level of backtracking. Note that if more than one level of backtracking occurs, the virtual forbidden region will also represent constraints due to link \( i + 1 \), \( i + 2 \) etc. The maximum level \( b \) of backtracking is a user definable parameter. Although it is probably possible to construct examples where full backtracking may be needed (\( n - 2 \) for \( n \)-DOF arm), our experimental results show that \( b \) is usually 2 or 3 for arms with up to 8 degrees of freedom.
Figure 4.5: Determining $t_i^{\text{block}}$ in the $t_i \times q_i$ space. (a) a conical region is swept along the straight line from start to goal, (b) search is limited to a diamond shaped region.
Figure 4.6: Backtracking in the $t \times q$ spaces, (a) and (b) are the bitmaps and paths of $t_2 \times q_2$ and $t_3 \times q_3$ before backtracking, (c) and (d) are the bitmaps and paths of $t_2 \times q_2$ and $t_3 \times q_3$ after backtracking.
4.2 Algorithmic Description of the Planner

Let \( n \) be the number of degrees of freedom of the manipulator and \( b \) be the maximum backtracking level allowed. An algorithmic description is as follows:

\[
\text{FIND_VALID_RANGES}(q_i);
\]

If ( \([q_i^l, q_i^u]\) is not a sub-interval of the valid ranges )

\[
\text{return}(\text{success, no path for link } 1);
\]

else

\[
\text{FORM_A_LINEAR_PATH_FOR}((t_i^u, q_i^u), (t_i^l, q_i^l));
\]

\[
\text{PARAMETERIZE}(Q_1, t_2);
\]

\[
i = 2; \ \text{blevel} = 1;
\]

while ( \( i \leq \text{DOF} \) )

\[
\text{BUILD_TQ_BITMAP}(t_i \times q_i);
\]

\[
\text{COMPUTE_L1_DISTANCE_MAP}(t_i \times q_i);
\]

\[
\text{COMPUTE_POTENTIAL_FIELD}(t_i \times q_i, (t_i^u, q_i^u));
\]

if (\( \text{HAS_POTENTIAL}((t_i^u, q_i^u)) \)) (i.e. path found )

\[
\text{PLAN_PATH}(t_i \times q_i);
\]

\[
\text{PARAMETERIZE}(Q_i, t_{i+1});
\]

else

while ( \( \text{blevel} \leq b \) )

\[
t_{i-\text{blevel}}^{\text{block}} = \text{DETERMINE_BLOCK_POSITION_T}(Q_{i-\text{blevel}});
\]

\[
(t_{i-\text{blevel}}^u, q_{i-\text{blevel}}^u) = Q_{i-\text{blevel}}(t_{i-\text{blevel}}^{\text{block}});
\]

\[
\text{PUT_VF_REGION}((t_{i-\text{blevel}}^u, q_{i-\text{blevel}}^u), Q_{i-\text{blevel}});
\]

\[
\text{new_path} = \text{PLAN_PATH}(t_{i-\text{blevel}} \times q_{i-\text{blevel}});
\]

if (\( \text{new_path} \))

\[
i = i - \text{blevel} + 1;
\]

\[
\text{blevel} = 1;
\]

\[
\text{PARAMETERIZE}(Q_i, t_{i+1});
\]

\[
\text{exit while}.
\]

else

\[
\text{blevel} = \text{blevel} + 1;
\]

end if
CHAPTER 4. BACKTRACKING STRATEGY

if (blevel > b)
    return (failure, max backtracking level exceeded)
end if

end while

end if

end while

return (success, path found)

end if

where,

PLAN_PATH(t, x, t;)
begin
Q, = FIND_PATH_BY_GRADIENT((t, , q, ), (t, , q, ), t, x, q, );
Q, = FORM_FORWARD_EXTENSION((t, , q, ), t, x, q, );
Q, = FORM_BACKWARD_EXTENSION((t, , q, ), t, x, q, );
Q, = CASCADE_PATHS(Q, , Q, , Q, );
end

Where:

FIND_VALID_RANGES(q, ) return the set of collision-free intervals for the first joint.

FORM A LINEAR PATH FOR((t, , q, ), (t, , q, )) returns a straight line segment connecting (t, , q, ) and (t, , q, ) with extension to the full range of the valid interval.

PARAMETERIZE(Q, t, t, 1) returns a parameterization of Q, .

BUILD_TQ_BITMAP(t, x, q, ) returns a bitmap for the t, x, q, space.

COMPUTE_L1_DISTANCE_MAP(t, x, q, ) returns an L1 distance map of t, x, q, space.

COMPUTE_POTENTIAL_FIELD(t, x, q, , t, , q, , ) returns a potential field for t, x, q, space starting from (t, , q, ).
CHAPTER 4. BACKTRACKING STRATEGY

FIND_PATH_BY_GRADIENT((t^i, q^i), (t^0, q^0), t_i \times q_i) returns a t \times q path in t_i \times q_i from (t^i, q^i) to (t^0, q^0) by a gradient search.

FORM_FORWARD_EXTENSION(((t^i, q^i), t_i \times q_i) returns a forward extension path from (t^i, q^i) in t_i \times q_i.

FORM_BACKWARD_EXTENSION(((t^i, q^i), t_i \times q_i) returns a backward extension path from (t^i, q^i) in t_i \times q_i.

CASCADE.Paths(Q_{ib}, Q_{im}, Q_{if}) returns a cascaded sequence of Q_{ib}, Q_{im}, and Q_{if}.

DETERMINE_BLOCK_POSITION_T(Q_{i+1}^{blevel}) returns a block t parameter where the block occurred in the t_{i+1} \times q_{i+1} space.

PUT_VF_REGION(((t^v_{i+1}, q^v_{i+1}), Q_{i+1}^{blevel}) puts a virtual forbidden region at (t^v_{i+1}, q^v_{i+1}) of the t_{i+1} \times q_{i+1} space.

4.3 Empirical Complexity Analysis

An exact complexity analysis of the new approach within the sequential framework is very difficult due to the discrete representation/search and many approximations in computations. This section will present an approximate and somewhat empirical analysis for our approach. The assumptions used in the analysis are based on the results of hundreds of tests for the implemented planners. A complexity analysis upon these assumptions will give a practical idea concerning the efficiency of the planner.

Assume a collision check for a link at a given configuration is a unit cost. Clearly, such a cost depends on the geometry of the robot link and the workspace obstacles, however, appropriate functions can be substituted in more exact analysis. The main aim here is to provide an estimate of the complexity associated with the sizes of the t_i \times q_i space and the backtracking level.

The planner complexity mainly depends on the complexity of building and searching the t_i \times q_i spaces. Experimentally, it is building the t \times q space bitmap that is
CHAPTER 4. BACKTRACKING STRATEGY

computationally intensive and dominates other computations such as building the potential field and searching for a path by about an order of magnitude. Assuming a constant time for a collision-check for a single link, the complexity of building the $t_i \times q_i$ bitmap is linear in the size of the $t_i \times q_i$ array. Let $\text{dim} t_i \times \text{dim} q_i$ be dimension of the discretized $t_i \times q_i$ space bitmap.

Let $f_i$ be the cost of building the $t_i \times q_i$ bitmap, $g_i$ be the cost of building the numerical potential fields over the bitmap, and $h_i$ be the cost of searching the $t_i \times q_i$ space for a path.

Clearly, the cost for planning for link $i$ (without any backtracking) is

$$ (f_i + h_i + g_i) $$

For level one backtracking, (say for link $i$), the planner re-builds an $L^1$ distance map and a potential field for link $i - 1$ (cost $h_{i-1}$), searches the $t_{i-1} \times q_{i-1}$ space (cost $g_{i-1}$), builds a new $t_i \times q_i$ bitmap (cost $f_i$), builds an $L^1$ map and a potential for link $i$ (cost $h_i$), and searches the $t_i \times q_i$ space (cost $g_i$). Furthermore assume that a maximum of $p_{i-1}$ backtrackings are allowed for link $i$. Thus the complexity for planning link $i$ with level 1 backtracking is

$$ p_{i-1}(h_{i-1} + g_{i-1} + f_i + h_i + g_i) + (f_i + h_i + g_i) $$

Similarly, for level 2 backtracking, at most $p_{i-2}$ level 2 backtrackings may be carried out, so the cost for planning for link $i$ at level 2 backtracking is

$$ p_{i-2}(h_{i-2} + g_{i-2} + f_{i-1} + p_{i-1}(h_{i-1} + g_{i-1} + f_i + h_i + g_i) + (f_i + h_i + g_i)) $$

Generalizing from above, the cost of planning for link $i$ with level $b$ backtracking is

$$ \text{cost}^b_{ii} = p_{i-b}(h_{i-b} + g_{i-b} + f_{i-b+1} + p_{i-b+1}(h_{i-b+1} + g_{i-b+1} + f_{i-b+2} + ... + p_{i-2}(h_{i-2} + g_{i-2} + f_{i-1} + p_{i-1}(h_{i-1} + g_{i-1} + f_i + h_i + g_i) + (f_i + h_i + g_i))...$$
CHAPTER 4. BACKTRACKING STRATEGY

56

Given that $f_i$ depends on the motion of all the previous $i-1$ links, the cost could be exponential in the worst case. Within our model, this complexity manifests itself in the increased size of the $t_i \times q_i$ spaces with $i$. Note that $dim t_i$ is always greater than or equal to $dim t_{i-1}$. We have empirically observed that the ratio $dim t_i / dim t_{i-1}$ varies between 1.0 and 2.5. Let $C$ be the upper bound for this ratio, so $dim t_i \leq C \times dim t_{i-1} \leq C^{i-1} \times dim t_1$. Because $f_i$ is proportional to $dim t_i \times dim q_i$, we have $f_i \leq C \times f_{i-1} \leq C^{i-1} \times f$, where $f_1 = f$.

Now, we make some assumptions based on our empirical observations. First, let $p_i$ be bounded by $p$, i.e., the maximum number of paths that are searched in a $t \times q$ space ($p \approx 25$ in the implemented planner). Let $f_1 = f$, $g_1 = g$, and $h_1 = h$. Further, we assume that $f_i$, $g_i$, and $h_i$ vary exponentially with $i$, i.e., $f_i = C^{i-1} f$, $g_i = C^{i-1} g$, and $h_i = C^{i-1} h$. Then,

$$\text{cost}_{li}^b = p(C^{i-b}h + C^{i-b}g + C^{i-b}f + p(C^{i-b}h + C^{i-b}g + C^{i-b+1}f + \ldots + p(C^{i-3}h + C^{i-3}g + C^{i-2}f + p(C^{i-2}h + C^{i-2}g + C^{i-1}f + C^{i-1}h + C^{i-1}g)) \ldots)$$

$$= \sum_{j=1}^{b} p^i C^{i-b+j-1}(h + g) + \sum_{j=1}^{b} p^i C^{i-b+j}f + p^{b-1}C^{i-1}(f + h + g)$$

Rearranging terms, we get

$$\text{cost}_{li}^b = \sum_{j=1}^{b} p^i C^{i-b+j-1}(h + g + Cf) + p^{b-1}C^{i-1}(f + h + g)$$

Therefore, the overall complexity of the planner for level $b$ backtracking is

$$\sum_{i=1}^{n} \text{cost}_{li}^b = \sum_{i=1}^{n} \sum_{j=1}^{b} p^i C^{i-b+j-1}(h + g + Cf) + \sum_{i=1}^{n} p^{b-1}C^{i-1}(f + h + g)$$

$$= (h + g + Cf) \sum_{i=1}^{n} \sum_{j=1}^{b} p^i C^{i-b+j} + p^{b-1}(f + h + g) \sum_{i=1}^{n} C^{i-1}$$

$$= (h + g + Cf)C^{b-1}(\sum_{i=1}^{n} C^{i}(\sum_{j=1}^{b} (pC)^j) + p^{b-1}(f + h + g)\frac{(C^n - 1)}{C - 1}$$

$$= (h + g + Cf)C^{b-1}(\frac{C(C^n - 1)}{C - 1})(\frac{(pC)((pC)^b - 1)}{(pC) - 1}) + p^{b-1}(f + h + g)\frac{(C^n - 1)}{C - 1}$$
Empirically speaking, since $f$ dominates the computations, $p \gg 1$ and $C \simeq 2$, an approximate and simplified expression for the overall complexity with $b$ level backtracking is

$$O(C^n p^b f).$$  \hfill (4.1)

There are two exponential factors, $C^n$ and $p^b$, in the complexity. In hundreds of path planning experiments with the planner, we observed that $C$ is a small constant which averages around 1.5, and $b$ is usually 2 or 3. As $C$ is small and every stage of the sequential planning is conducted very quickly in a 2-dimensional subspace, the implemented planner is still feasible for manipulators with large numbers of DOFs, for example, the SPDM space arm with 7 DOFs. Chapter 5 presents experimental results with a variety of implemented planners that show the practicality of our approach.
Chapter 5

Experimental Results and Planner Evaluation

Four examples are given in this chapter to illustrate the capacity of the new bitmap-based approach to deal with various manipulators in different environments. In order to illustrate the efficiency of the new approach, a series of experiments have been conducted. The experimental results provide the basis for the assumptions in the complexity analysis in the previous chapter.

5.1 Examples on Several Manipulators

5.1.1 A Simple 4-DOF Arm in 3-D Workspace

The first example shown in Figure 5.1 is chosen to illustrate the planner maneuvering to plan a collision-free path in the presence of obstacles. A 4-DOF manipulator with revolute joints with joint limits of $\pm 180$ degree on all four joints is placed in an environment with an “H”-shape obstacle. The start and goal positions are deliberately chosen in the two “U”-shaped traps of the obstacle. To accomplish the task, the manipulator has to move back out of the left “U” first, then move into the right “U” in an intelligent manner to accomplish the task. The planner successfully found
Figure 5.1: An example of the planned motion sequences for a 4-DOF manipulator.
a path in 172 seconds on an SGI Indigo II machine at a 2-degree resolution for all joints. Note that planner had no backtrackings for this example. The planned path is sampled in Figure 5.1 from 1 to 8. The start and goal positions are in Figure 5.1 (1) and (8) respectively.

The $t \times q$ spaces for this example are shown in Figure 5.2 from left to the right. As the workspace is simple, the $t \times q$ spaces are very sparse. The $t_1 \times q_1$ and $t_2 \times q_2$ spaces are completely free. However, the $(t^*_3, q^*_3)$ and $(t^*_3, q^*_3)$ are not directly reachable due to the presence of the middle bar of the "H" obstacle. This middle bar is shown as a narrow forbidden region between the $(t^*_3, q^*_3)$ and $(t^*_3, q^*_3)$ in the $t_3 \times q_3$ space. The path connecting the $(t^*_3, q^*_3)$ and $(t^*_3, q^*_3)$ corresponds to the obstacle-avoidance motion for the first three links. There is a narrow free-space channel near the $(t^*_4, q^*_4)$ in the $t_4 \times q_4$ space, which reflects the left "U" trap of the obstacle. The vertical downward section starting from $(t^*_4, q^*_4)$ means that link 4 first backs away in order to obtain a collision-free motion toward the goal $(t^*_4, q^*_4)$.

Notice that all paths in the $t \times q$ spaces tend to be in the center of the free space, this property is due to the Voronoi-based potential fields. The final path for the manipulator is a safe path in the workspace. The safe paths sometimes cause the manipulator to move too far from the obstacles, especially in a very sparse workspace. However, a simple path optimization strategy can easily rectify the path for a given criterion.
Table 5.1: Detailed run times for a 4-DOF arm.

<table>
<thead>
<tr>
<th>(in seconds)</th>
<th>TQ bitmap</th>
<th>$L_1$ distance</th>
<th>Potential</th>
<th>Searching</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link 1</td>
<td>0.080</td>
<td>N.A.</td>
<td>N.A.</td>
<td>0.060*</td>
<td>0.140</td>
</tr>
<tr>
<td>Link 2</td>
<td>2.880</td>
<td>2.840</td>
<td>0.770</td>
<td>0.000</td>
<td>6.490</td>
</tr>
<tr>
<td>Link 3</td>
<td>62.861</td>
<td>4.060</td>
<td>1.180</td>
<td>0.010</td>
<td>68.101</td>
</tr>
<tr>
<td>Link 4</td>
<td>89.091</td>
<td>2.710</td>
<td>1.010</td>
<td>0.010</td>
<td>92.821</td>
</tr>
<tr>
<td>Total</td>
<td>154.912</td>
<td>9.610</td>
<td>2.960</td>
<td>0.070</td>
<td>167.552</td>
</tr>
</tbody>
</table>

* Link 1 is a special case for which there is no distance map, potential field.

The $t \times q$ space sizes in pixels are $180 \times 180$, $180 \times 180$, $270 \times 180$, and $520 \times 180$ for $t_1 \times q_1$ to $t_4 \times q_4$. This gives an average rate of increase for $C \ (520/180)^{1/3} = 1.4242$. Table 5.1 gives a detailed run-times for the planner in every stage of planning, i.e., building the $t_i \times q_i$ bitmap, computing the $L_1$ distance map, computing the potential field, and searching for a $t_i \times q_i$ path. Note that the building the $t_i \times q_i$ bitmap is the most time consuming step in the planning process. For this example, the $t_i \times q_i$ bitmap computation took 92% of time.

5.1.2 A 6-DOF Arm in 2-D Workspace

A 6-DOF example in 2-dimensional workspace is shown in Figure 5.3 and 5.4. All joints are revolute. The first link of the arm has a joint limit of $\pm 180$ degree, and the rest of links have $\pm 135$ degree limits. In this example, there are 6 square obstacles in the workspace of the planar arm. The start position is at the left configuration in Figure 5.3, and the goal position is at the right configuration. The goal configuration is designed to be difficult for the 6-DOF arm to reach due to joint limits and the self-collision constraints.

In this example, 15 backtrackings were observed at level 1 for link 5. The planned motion of the arm was very complicated in order to achieve the goal. The arm had to extend to the wider free opening on the upper right corner first, then shrink the last four links together to squeeze through the narrow channel near the goal position. The winding path is also reflected in the $t \times q$ spaces in Figures 5.5. Figure 5.6 is an enlargement of the $t_5 \times q_5$ space in Figures 5.5. The tooth-like path represents many
to-and-fro motions for link 5.

Due to the 15 backtrackings for link 5, it took about 60 minutes for this example on a SUN SPARC 10 station. In this example, the sizes of the $t \times q$ spaces are $360 \times 360$, $360 \times 360$, $1203 \times 360$, $1876 \times 360$, $2701 \times 360$, and $3137 \times 360$ respectively. The average $C$ is computed as 1.3624.

5.1.3 An 8-DOF Arm in 2-D Workspace

The example in Figure 5.7 is a planar 8-DOF arm, which is similar to the one used by Barraquand and Latombe (Barraquand and Latombe 1991). This example shows the capability of our new approach in dealing with fairly large numbers of degrees of freedom. The first joint has no limit, and the remaining 7 joints have $\pm 170$ degree limits. The start position of the arm is placed in a “shrunk” state at the bottom. The goal position extends to the top-right of the workspace. The “door” between two horizontal bars is fairly narrow considering the restrictions of link lengths, joint limits

Figure 5.3: An example of a 6-DOF manipulator. The initial (left configuration) and final (right configuration) positions workspace for an example of a 6-DOF manipulator in the 2-D workspace.
CHAPTER 5. EXPERIMENTAL RESULTS AND PLANNER EVALUATION

Figure 5.4: The continuous motion for a 6-DOF manipulator in a 2-D workspace.

Figure 5.5: The \( t \times q \) spaces of a 6-DOFs manipulator. The \( t_1 \times q_1 \) to \( t_6 \times q_6 \) spaces are from top to bottom and from left to right. The sizes are 360×360, 360×360, 1203×360, 1876×360, 2701×360, and 3137×360 respectively.
and self-collision.

For this example, amazingly, only one backtracking at link 8 was observed during the simulation. This illustrates that the new planner often requires very few backtrackings even for quite large numbers of DOFs. Figure 5.8 shows the motion planned for the task. The arm shrinks back to the right first, then extends the arm link by link through the narrow "door" moving link 8 first. Obviously, such a motion for the arm is an intelligent choice in this situation. On a SUN SPARC 10 station, this example took 24 minutes and 22 seconds of CPU time.

Figure 5.9 contains the $t \times q$ spaces for this example. The sizes of these $t \times q$ spaces are $360 \times 360$, $360 \times 360$, $847 \times 360$, $1952 \times 360$, $2179 \times 360$, $2917 \times 360$, $3499 \times 360$, and $3023 \times 360$ respectively. Certainly the sizes likely increases link by link, however the average $C$ in this example is only 1.3552. Such an increase is acceptable for a fairly large number of DOFs (we believe that it would work for 10-12 DOFs fairly reasonably).
CHAPTER 5. EXPERIMENTAL RESULTS AND PLANNER EVALUATION

Figure 5.7: An example of an 8-DOF manipulator in a 2-D workspace. The start is the left configuration and goal is the right configuration.

Figure 5.8: The continuous motion for an 8-DOF manipulator in 2-D workspace.
CHAPTER 5. EXPERIMENTAL RESULTS AND PLANNER EVALUATION

Figure 5.9: The $t \times q$ spaces of an 8-DOF manipulator. The $t_1 \times q_1$ to $t_8 \times q_8$ spaces are from top to bottom and from left to right. The sizes are $360 \times 360$, $360 \times 360$, $847 \times 360$, $1952 \times 360$, $2179 \times 360$, $2917 \times 360$, $3499 \times 360$, and $3023 \times 360$ respectively.

5.1.4 A 7-DOF Example in 3-D Workspace

The implemented motion planner has successfully been applied to a STM \(^1\) arm (STM stands for SPDM Testbed Manipulator) built by International Submarine Ltd, which has been proposed to be used in the space station. It has 7 degrees of freedom (Figure 5.10). The first 2 links (links 1 and 2) are relatively short. This makes the first three joints act like a joystick. Links 3 and 4 are longer to give the space arm a large operation space. The arm and obstacles are modeled using volumetric primitives and their unions. The volumetric primitives include parallel pipeds, cylinders, cut cylinders, cones, rounded cones, spheres, ellipsoids etc. These primitives may be grouped to represent very complicated shapes of objects. In this simulation of application, a 7-piece support structure is used to fix the space arm onto a square platform. Note that the support and platform are also considered as obstacles in motion planning process.

\(^{1}\)Courtesy of International Submarine Engineering Ltd.
Figure 5.10: The SPDM testbed manipulator (STM).
CHAPTER 5. EXPERIMENTAL RESULTS AND PLANNER EVALUATION

This example has been tested in various situations, and the run times are approximately 5 minutes to 20 minutes on an SGI IRIS Indigo 2. An example of the STM arm is shown in Figure 5.11. A “T”-shaped obstacle, two vertical cylinders and the supporting structure comprise the working environment for the space arm. The start and the goal positions are chosen on the opposite side of the “T”-shaped obstacle. (Figure 5.11 (1) and (8) respectively.) The freedom of motion near the start and goal positions and in between the supporting structure and the other obstacles is very limited.

The resolution for this example is at 2 degrees for all joints. It took 730 seconds for the planner to find a path for this difficult task. The $t \times q$ spaces are shown in Figure 5.13. The sizes of the $t \times q$ spaces are 180×180, 180×180, 268×180, 288×180, 509×180, 540×180, and 716×180 respectively. Thus the average increase $C$ is as small as 1.2588. We observed that only 3 backtrackings were needed at link 7.

Table 5.2 gives detailed run-times for this 7-DOF example. Again, building the $t_i \times q_i$ bitmap is the dominant in the computational cost. For this example, 89% of the whole planning time was spent on bitmap computation.

Notice that some of the $t \times q$ spaces are very sparse and there are no forbidden regions (other than those imposed by joint limits, shown as the top and bottom parallel bars presented in some $t \times q$ spaces). In the earlier V-Graph implementation, if backtracking occurred to a $t_i \times q_i$ space which is sparse, in order to give a set of possible paths for the V-Graph search, an ad hoc mechanism of adding nodes had to be invoked. The new approach avoids this inconvenience. The systematic backtracking strategy using the virtual forbidden regions provides a uniform mechanism to search the $t \times q$ spaces.

5.2 Experiments

Two factors, $p^b$ and $C^n$, dominate the complexity for our new approach. Since the conducted experiments show that $b$ is usually less than 3 and $C$ is approximately 1.5 or less, there are still a few questions concerning these two factors. How many backtrackings are practically needed to be accomplished? How does the maximum
Figure 5.11: An example of the planned motion sequences for a 7-DOF manipulator. (Continued in next figure.)
Figure 5.12: An example of the planned motion sequences for a 7-DOF manipulator. (Continuing of last figure.)
Figure 5.13: The $t \times q$ spaces of a 7-DOFs manipulator. The $t_1 \times q_1$ to $t_7 \times q_7$ spaces are from top to bottom and from left to right. The sizes are 180x180, 180x180, 268x180, 288x180, 509x180, 540x180, and 716x180 respectively.

Table 5.2: Detailed run times for a 7-DOF manipulator.

<table>
<thead>
<tr>
<th>(in seconds)</th>
<th>$t \times q$ bitmap</th>
<th>L1 distance</th>
<th>Potential</th>
<th>Searching</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link 1</td>
<td>0.130</td>
<td>N.A.</td>
<td>N.A.</td>
<td>0.050*</td>
<td>0.180</td>
</tr>
<tr>
<td>Link 2</td>
<td>5.480</td>
<td>1.850</td>
<td>0.570</td>
<td>0.010</td>
<td>7.910</td>
</tr>
<tr>
<td>Link 3</td>
<td>106.351</td>
<td>2.340</td>
<td>0.660</td>
<td>0.010</td>
<td>103.361</td>
</tr>
<tr>
<td>Link 4</td>
<td>127.791</td>
<td>3.860</td>
<td>1.400</td>
<td>0.020</td>
<td>133.071</td>
</tr>
<tr>
<td>Link 5</td>
<td>24.630</td>
<td>6.760</td>
<td>2.080</td>
<td>0.020</td>
<td>33.430</td>
</tr>
<tr>
<td>Link 6</td>
<td>47.990</td>
<td>7.560</td>
<td>2.580</td>
<td>0.020</td>
<td>58.150</td>
</tr>
<tr>
<td>Link 7</td>
<td>85.131</td>
<td>4.170</td>
<td>0.360</td>
<td>N.A.</td>
<td>89.661</td>
</tr>
<tr>
<td>Link 8</td>
<td>0.070</td>
<td>7.330</td>
<td>2.560</td>
<td>0.020</td>
<td>9.980</td>
</tr>
<tr>
<td>Link 7</td>
<td>90.021</td>
<td>4.210</td>
<td>0.380</td>
<td>N.A.</td>
<td>94.611</td>
</tr>
<tr>
<td>Link 6</td>
<td>0.050</td>
<td>7.310</td>
<td>2.540</td>
<td>0.010</td>
<td>9.910</td>
</tr>
<tr>
<td>Link 7</td>
<td>89.181</td>
<td>4.060</td>
<td>0.380</td>
<td>N.A.</td>
<td>93.621</td>
</tr>
<tr>
<td>Link 6</td>
<td>0.060</td>
<td>7.290</td>
<td>2.580</td>
<td>0.020</td>
<td>9.950</td>
</tr>
<tr>
<td>Link 7</td>
<td>79.091</td>
<td>4.490</td>
<td>1.230</td>
<td>0.010</td>
<td>84.821</td>
</tr>
<tr>
<td>Total</td>
<td>649.976</td>
<td>61.230</td>
<td>17.320</td>
<td>0.190</td>
<td>728.716</td>
</tr>
</tbody>
</table>

* Link 1 is a special case for which there is no distance map, potential field.
backtracking level \( b \) affect the success rate of the planner? How does the exponential factor \( C^n \) affect the efficiency of the planner? With these questions in mind, a series of extensive experiments have been designed and conducted for different manipulators with DOF numbers varying from 4 to 8.

The first set of the experiments was conducted for planar manipulators. For a given manipulator, several obstacles were randomly placed in its workspace. The number of obstacles was fixed for a given manipulator. The start and goal configurations were also randomly selected, and a collision detection mechanism was used to reject any invalid situation where either of the configurations intersected with the obstacles or the manipulator collided with itself.

The manipulator arms had revolute joints with 4 to 8 degrees of freedom. The first joint had no joint limit, while rest of the joints were limited to \( \pm 135 \) degrees. Although the obstacles were squares of a fixed dimension, the random placement of these simple forms easily gave rise to environments of quite arbitrary complexity. Hence, using simple shapes was not a limitation at all.

For a 4-DOF manipulator similar to the one used in Section 5.1.1, the number of obstacles was 10, the maximum backtracking level was 2, and the maximum backtracking number in a level was 20 (this limit of 20 was never reached). For a 6-DOF manipulator, the number of obstacles was 6, the maximum backtracking level was 3, and the maximum backtracking number in a level was 20.

More than a hundred examples for each manipulator have been tested. Some typical results are shown in Tables 5.3, 5.4, 5.5 and 5.6.

The second set of experiments was also carried out in a 2-D workspace and used the same environment with 10 obstacles. The start and goal configurations were once again randomly selected. Table 5.7 shows the experimental statistics for a 4-DOF arm.

The third set of experiments was conducted for the 7-DOF STM arm. The environments are similar to the one shown in Figure 5.11 with different start and goal positions that were manually selected. Table 5.8 lists the run times for some typical runs. Table 5.9 gives the statistics on average run and backtracking level. The
Table 5.3: Examples of run times for the four link manipulator arm, where 10 obstacles were randomly placed in the 2-D workspace. The maximum backtracking level was set at 2. In column 3, $x(i)$ denotes that planner backtracked $x$ number of times for link $i$.

<table>
<thead>
<tr>
<th>Actual Backtracking Level</th>
<th>CPU Time in Seconds</th>
<th>Number of Backtrackings Number(link)</th>
<th>Path Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>179.1</td>
<td>1(3)</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>372.0</td>
<td>1(3)2(4)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>111.3</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>1181.6</td>
<td>1(3)3(4)1(3)3(4)2(3)2(4)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>187.0</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>139.4</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>307.0</td>
<td>1(3)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>159.1</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>233.9</td>
<td>1(3)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>184.3</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>0</td>
<td>174.7</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>140.1</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>123.7</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>130.3</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>197.2</td>
<td>1(4)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>136.4</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>207.9</td>
<td>1(4)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>223.3</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>7.3</td>
<td>0</td>
<td>no path</td>
</tr>
</tbody>
</table>

Table 5.4: Numerical results for the four link manipulator, where obstacles were randomly positioned in its environment, and the number of obstacles was 10.

<table>
<thead>
<tr>
<th>Maximum Level of Backtracking</th>
<th>Average CPU Time in Seconds</th>
<th>Average Number of Paths Searched</th>
<th>Percentage of Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>130.9</td>
<td>4.0</td>
<td>77.8</td>
</tr>
<tr>
<td>1</td>
<td>155.7</td>
<td>4.2</td>
<td>98.9</td>
</tr>
<tr>
<td>2</td>
<td>167.1</td>
<td>4.4</td>
<td>100.0</td>
</tr>
</tbody>
</table>
Table 5.5: Examples of run times for the six link manipulator, where 6 obstacles were randomly placed. The maximum backtracking level was set at 3. In column 3, $x(i)$ denotes that the planner backtracked $x$ number of times for link $i$.

<table>
<thead>
<tr>
<th>Actual Backtracking Level</th>
<th>CPU Time in Seconds</th>
<th>Number of Backtrackings Number(link)</th>
<th>Path Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3527.8</td>
<td>3(5)4(4)3(5)1(4)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>414.9</td>
<td>1(3)3(5)6(4)3(5)</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>597.5</td>
<td>34(2)3(1)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>357.2</td>
<td>1(3)1(6)</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>605.7</td>
<td>3(6)1(5)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>380.4</td>
<td>0</td>
<td>no path</td>
</tr>
<tr>
<td>0</td>
<td>5.5</td>
<td>2(6)</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 5.6: Numerical results for the six-link manipulator, where obstacles were randomly positioned in its environment.

<table>
<thead>
<tr>
<th>Maximum Level of Backtracking</th>
<th>Average CPU Time in Seconds</th>
<th>Average Number of Paths Searched</th>
<th>Percentage of Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>261.0</td>
<td>6.0</td>
<td>84.4</td>
</tr>
<tr>
<td>1</td>
<td>343.1</td>
<td>6.4</td>
<td>97.4</td>
</tr>
<tr>
<td>2</td>
<td>346.6</td>
<td>6.4</td>
<td>98.7</td>
</tr>
<tr>
<td>3</td>
<td>387.9</td>
<td>6.8</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Table 5.7: Numerical results for the four link manipulator with 10 fixed obstacles.

<table>
<thead>
<tr>
<th>Maximum Level of Backtracking</th>
<th>Average CPU Time in Seconds</th>
<th>Average Number of Paths Searched</th>
<th>% of Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>101.6</td>
<td>4.0</td>
<td>32.7</td>
</tr>
<tr>
<td>1</td>
<td>277.9</td>
<td>5.5</td>
<td>97.1</td>
</tr>
<tr>
<td>2</td>
<td>303.2</td>
<td>5.8</td>
<td>100.0</td>
</tr>
</tbody>
</table>
Table 5.8: Examples of run times for a 7-DOF manipulator in 3-dimensional workspaces.

<table>
<thead>
<tr>
<th>Actual Backtracking Level</th>
<th>Clock Time in Seconds</th>
<th>Number of Backtrackings Number(link)</th>
<th>Path Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>323</td>
<td>1(7)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>245</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>204</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>1062</td>
<td>7(7)</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>345</td>
<td>1(4)</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>321</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>267</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>534</td>
<td>4(7)</td>
<td>yes</td>
</tr>
</tbody>
</table>

maximum backtracking level required was 1 for these sets of cases.

Theoretically the planner is not complete, however, at least in the hundreds of experiments conducted, it succeeded in 100 percent of the cases with a maximum backtracking level of 2 for the 4-DOF planar example, at 3 for the 6-DOF planar example, and at 1 for the 7-DOF 3-D space arm. With a fairly high resolution — one degree for all DOFs for the planar case and two degrees for the 3-dimensional case — the average run times are approximately 3 minutes for the 4-DOF manipulator, 7 minutes for the 6-DOF manipulator, and 8 minutes for the 7-DOF STM arm. In most cases, single level backtracking is sufficient to solve the problem. The success rates with single level backtracking are above 95% for the three sets of experiments. The very first example in Table 5.5 is an extreme case where the planner took 30 backtrackings with the maximum level of backtracking at 3. However, such cases are extremely rare.

The constant $C$ was observed to be an acceptable small number around 1.5. For example, the $C$’s for the four examples in the Section 5.1 are approximately 1.42, 1.36, 1.36, and 1.26 respectively. Because $t \times q$ spaces are 2-dimensional, the memory requirement for fairly large numbers of DOFs is still quite moderate with such a rate of increase. For example, the memory size for $t_8 \times q_8$ space bitmap of the 8-DOF arm at one degree resolution is roughly 256K bytes only. Hence, practically speaking, the
Table 5.9: Average run times for the 7-DOF manipulator in 3-dimensional workspaces.

<table>
<thead>
<tr>
<th>Maximum Level of Backtracking</th>
<th>Average Clock Time in Seconds</th>
<th>Average Number of Paths Searched</th>
<th>% of Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>261.4</td>
<td>7.0</td>
<td>50.0</td>
</tr>
<tr>
<td>1</td>
<td>480.0</td>
<td>9.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Table 5.10: Numerical results for a V-Graph based planner for a six link manipulator in 3-dimensional workspace. Obstacles were randomly placed in its environment, and the number of obstacles was varied randomly between 4 and 10.

<table>
<thead>
<tr>
<th>Maximum Level of Backtracking</th>
<th>Average Clock Time in Seconds</th>
<th>Average Number of Paths Searched</th>
<th>% of Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1369</td>
<td>6</td>
<td>37</td>
</tr>
<tr>
<td>1</td>
<td>2715</td>
<td>8</td>
<td>73</td>
</tr>
<tr>
<td>2</td>
<td>2831</td>
<td>11</td>
<td>91</td>
</tr>
</tbody>
</table>

factor $C$ is acceptable for a fairly large number of DOFs. Admittedly, the results are empirical, however, we believe they demonstrate that the new approach is practical.

A high percentage of success rates with few backtrackings at a very small level has been achieved using the new bitmap-based approach. This shows the efficiency of the $t \times q$ path choice and the backtracking mechanism has been significantly improved. In most cases, one level backtracking is sufficient to solve the problem. Considering that the maximum backtracking level was always within 3, and the maximum number of backtrackings never reached the limit of 20 during the experiments, the exponential factor $p^b$ is also acceptable in practice.

Table 5.10 shows statistic results for tests conducted using the previous V-Graph based implementation for a 6-DOF manipulator (Gupta and Zhu 1993). The manipulator was in a 3-dimensional workspace, however, the manipulator's movement was restricted to a planar surface because all the revolute joints were in same direction. Viewing the workspace from the front, the situation is very similar to the 6-DOF
example in Figure 5.4. The obstacles were also placed randomly. The number of obstacles was between 4 and 10. The results were obtained on a SPARC I station. The success rates reached 73% and 91% at one and two levels backtracking. The new bitmap-based approach improves the success rates significantly. Although the run times are not directly comparable due to the differences in arms and workspaces, the new approach seems to be much faster. Most importantly, the new approach, on average, leads to a fewer number of backtrackings at a given level and fewer levels of backtracking, and is practical for motion planning for manipulators with fairly large numbers of degrees of freedom.
Chapter 6

Conclusions

6.1 Summary

This thesis presents a new approach within the sequential framework for developing motion planners for manipulators with many degrees of freedom. The planner implemented using the new approach has been extensively tested with experiments for different manipulators with up to 8 degrees of freedom. The results show that, using the sequential search framework, the new approach is efficient and applicable for manipulators with fairly large numbers of degrees of freedom. This conclusion is firmly supported by two aspects of the new approach — bitmap-based potential-guided search — which solves the 2-dimensional sub-problems in an efficient manner, and the systematic backtracking mechanism which performs an efficient search of the n-dimensional solution space by using the concept of a virtual forbidden region.

The complexity of the planner implemented with the new approach is exponential with the dimensionality $n$ and the maximum backtracking level $b$, however, extensive experiments have shown that the base $C$ of the first exponential factor in the complexity is a small number around 1.5 and the maximum backtracking level $b$ is less than or equal to 3.

Although the approach within the sequential framework is incomplete, we have found it difficult to find an example to beat the implemented planner. At least during
CHAPTER 6. CONCLUSIONS

the hundreds of experiments conducted for a variety of manipulator arms, not a single failure was found because of the incompleteness of the planner. The planner has been demonstrated effective and efficient for a variety of manipulators with up to 8 degrees of the freedom. The average run times are within 10 minutes. In summary, the planner based on the new approach has a number of advantages:

- It avoids the difficulties of representing n-dimensional C-space obstacles and of searching the n-dimensional space.

- Although its complexity is exponential with the dimensionality \( n \), the constants are small (around 1.5).

- It is efficient and practical because the systematic search allows the planner to succeed with a limited number of backtrackings at very few levels.

- It is deterministic (i.e., no stochastic techniques are involved).

- It plans paths that tend to lie in the “middle” of the free-space.

6.2 Future Work

This section discusses future extensions and some interesting issues.

1. Backtracking mechanism

The way to choose the position of a virtual block region in the \( t_{i-1} \times q_{i-1} \) space is somewhat arbitrary. Although repeated backtracking has succeeded in most cases — even with such a simple choice of the position of virtual forbidden region — it may lead to unnecessary numbers of backtrackings, and even failures.

By introducing the notion of a virtual forbidden region, the backtracking mechanism leads to a systematic search for the \( t \times q \) space and consequently of the whole \( n \)-dimensional configuration space. Further theoretical and empirical work is needed to better understand the backtracking mechanism. Theoretical analysis of the relationship between the algorithmic complexity and the \( t \times q \) spaces’ geometrical properties would be very useful for future improvement.
2. Parallel processing

A natural solution to speed up the computation for the approach is parallel processing. Because the the representation and search in the $t \times q$ space is bitmap-based, a parallel implementation is a straightforward extension. Ideas for computing the C-space (Lozano-Pérez and O'Donnell 1991) should be very helpful for the computation of forbidden regions in the $t \times q$ spaces. The wavefront algorithms in building the numerical potential field in $t \times q$ space is very suitable for parallel computing. Note that about 90 percent of time consumption in the whole planning process is in building the $t \times q$ bitmaps (See Table 5.1 and Table 5.2), therefore a parallel implementation for this part will speed up the planning significantly. In addition, parallel search algorithms (Kumar and Rao 1987; Kumar, Ramesh, and Rao 1988; Challou, Gini, and Kumar 1993) could possibly be adapted directly.

3. Solving sub-problem with more than one degree of freedom

This implementation decomposes a $n$-dimensional motion planning problem into a series of 2-dimensional problems, i.e. planning one degree of freedom in each stage. However, in some cases, one may wish to plan a few degrees of freedom together to adapt to different manipulator structures or for other reasons, e.g., some manipulators may have more than one degree of freedom at one joint. For example, a spherical joint has three degrees of rotational freedom. In some cases, planning a few degrees of freedom together may be more efficient. For a mobile robot, one would like to plan the two Cartesian degrees of freedom together. Theoretically, there is no difficulty to implement the approach for the multi-DOF joint case.

4. Completeness

The implemented planner is not complete because the virtual forbidden region added during the backtracking may later lead to the loss of a set of possible paths. Reducing the size of the virtual forbidden region makes the situation better, but even a 1-pixel-size virtual forbidden region may not guarantee completeness. That is because determining the right blocking location is difficult
due to the diversity of the shapes of the $t \times q$ forbidden regions. If the block location is correctly determined, that is, the virtual forbidden region corresponds to a real deemed-failure configuration set, then the planner might be resolution-complete with a 1-pixel-size virtual forbidden region. For example, a block location corresponding a vertical strip shape of block is a right one (see Figure 4.4.(b)), but for complicated shape, determining the right block location will be difficult. A theoretical analysis based on topology could be the direction for future completeness explanation.
Appendix A

Planner System and Representations

A.1 Link Coordinate Systems

This thesis uses the standard notation similar to (Fu, Gonzalez, and Lee 1987) to describe each link of a manipulator arm. The robot and obstacle coordinate systems are all respect to a world frame $F_W$. The obstacle frame $F_O$ is related to the world frame by a position vector $p_o$. The robot frame $F_R$ is related to the world frame by a position vector $p_r$. The Denavit-Hartenberg (D-H) representation (Denavit and Hartenberg 1955), which results in a $4 \times 4$ homogeneous transformation matrix representing the each link's coordinate system with respect to the previous link's coordinate system, is used to describe the translational and rotational relationship between adjacent links.

Link frame $F_i$ is attached rigidly to link $i$. Thus the $n$-th coordinate frame moves with the end effector (link $n$). Every coordinate frame is established based on three rules:

1. The $z_i$ axis lies along the axis of motion of the $i$th joint.

2. The $x_i$ axis is normal to the $z_{i-1}$, and pointing away from it.
3. The $y_i$ axis is determined by the right-hand rule.

Once the coordinate system is established by following above rule, the relationship between adjacent links, i.e., the adjacent frames, will depend on the following four parameters:

- $a_i$: the distance from axis $z_{i+1}$ to axis $z_i$ measured along axis $x_{i+1}$;
- $d_i$: the distance from axis $x_i$ to axis $x_{i+1}$ measured along axis $z_i$;
- $\alpha_i$: the angle between axis $z_i$ and axis $z_{i+1}$ measured about axis $x_i$;
- $q_i$: the angle between axis $x_i$ and axis $x_{i+1}$ measured about axis $z_i$;
Figure A.2: Robot Coordinate Systems.
Figure A.1 shows the definitions of these parameters. Figure A.2 illustrates such a coordinate system. A homogeneous transformation composed by these four parameters represents the relationship between adjacent link frames, i.e.

\[
A_i^{i-1} = \begin{bmatrix}
\cos q_i & -\cos \alpha_i \sin q_i & \sin \alpha_i \sin q_i & a_i \cos q_i \\
\sin q_i & \cos \alpha_i \cos q_i & -\sin \alpha_i \cos q_i & a_i \sin q_i \\
0 & \sin \alpha_i & \cos \alpha_i & d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (A.1)

Hence a manipulator arm can be represented kinematically by giving the values of four parameters for each link. A homogeneous transformation

\[
T_i^0 = A_i^0 A_{i+1}^0 ... A_n^0 
\]

specifies the location of the ith coordinate frame with respect to the base coordinate system, i.e. the forward kinematics computation. Then the geometrical relation of the manipulator links and obstacle can be evaluated for the purpose of distance computation and collision detection.

## A.2 Planner Data Structures

### A.2.1 Data Structures for a Robot Arm

The robot arms in this thesis are considered to be a serial, non-branching chain of rigid links. In the 2-dimensional prototype of the sequential planner, each link of a robot arm is modeled as a rectangle, and obstacles are modeled as a set of rectangles. Such a simplification kept us away from complicated geometrical representations, since our goal is to experiment with various ideas for the sequential framework.

In the 2-dimensional implementation, the following data structures are used to represent a robot arm and obstacles.

1. Data Structures for Arm and Links
typedef struct arm
{
    int nlinks;
    Link *link;
} Arm;

where,

- nlinks: the number of links of a robot arm.
- link: a reference to the nlinks link of the arm.

typedef struct link
{
    float a, l;
    int nverts;
    FloatPoint base;
    FloatPoint vertex[MAX_VERTEX];
} Link;

where,

- a: the \( a_i \) defined in previous section.
- l: the actual link length.
- nverts: the number of vertices in the link model. (It is always 4 in the 2-D planner.)
- base: The coordinates of the axis of this link with respect to previous link’s coordination system.
- vertex[MAX_VERTEX]: an array of coordinates for the vertex of the link.

(2) Data Structures for Obstacles
typedef struct obstacle
{
    int nverts;
    float rot;
    FloatPoint trans;
    FloatPoint vertex[MAX_VERTEX];
} Obstacle;

where,

- nverts: the number of vertices of the obstacle model. (It is always 4 in the 2-D planner.)
- rot: the angle of rotation of the rectangle obstacle in radian.
- trans: the translation of the rectangle obstacle with respect to the global coordinate system.
- vertex[MAX_VERTEX]: vertices that describe the rectangle.

Then all obstacles are kept in a list for easy access.

typedef struct obstaclelist
{
    int nobss;
    Obstacle *obs;
} ObstacleList;

where,

- nobss: the number of total rectangle obstacles.
- obs: the list of all obstacles.
The 3-dimensional implementation of the sequential framework uses the ACT programming and modeling facilities (as we mentioned in Chapter 3), so the data structures for a robot arm and obstacles are part of internal hierarchical representation of the ACT modeling system. (Please see (Mazer and et al 1991; ALEPH 1992c; ALEPH 1992a; ALEPH 1992b) for details.)

### A.2.2 Data Structures for $t \times q$ Space and Paths

(1) Data Structure for $t \times q$ Spaces

There are mainly four bitmaps for each link, i.e., obstacle/free-space bitmap, $L^1$ distance map, Voronoi diagram map, and potential map. All four maps are kept in one structure as follows.

```c
typedef struct tqlinkcspace
{
    int ddimt, dimq;
    int t_bound, q_bound, p_bound;
    unsigned char *bitmap;
    short int *distmap;
    unsigned char *voro;
    short int *potential;
} TQLinkCSpace;

typedef struct tqcspace
{
    int nlinks;
    TQLinkCSpace *tqlink;
} TQCSpace;
```

where,

- \texttt{ddimt}: the dimension of \texttt{t} of the $t_i \times q_i$ space.
• dimq: the dimension of $q$ of the $t_i \times q_i$ space.

• t-bound, q-bound, p-bound: Working parameters that define whether bounds exist for the $t$, $q$, and $t_i \times q_i$ paths.

• bitmap: a 2-dimensional array of unsigned char for for storing obstacle/free-space bitmap of the $t_i \times q_i$ space.

• distmap: a 2-dimensional array of short integer storing the $L^1$ distance map of the $t_i \times q_i$ space.

• voro: a 2-dimensional array of unsigned char for storing the Voronoi diagram of the $t_i \times q_i$ space.

• potential: a 2-dimensional array of short integer for storing the potential map of the $t_i \times q_i$ space.

• nlinks: the number of total link of a robot arm, i.e., the degree of freedom.

Note all bitmaps are allocated and de-allocated dynamically in order to reduce memory usage requirement.

(2) Data Structure for $t \times q$ Paths

The coordinates of $t \times q$ space are denoted as $t$ and $q$. A $t \times q$ path is an array of $t \times q$ coordinates. Their structures are defined as follows.

```c
typedef struct tqconfig
{
    int t, q;
} TQConfig;

typedef struct linkpath
{
    int        dim;
    TQConfig   *d;
} LinkPath;
```
where,

- $\text{dim}$: the dimension of the $t \times q$ path.
- $d$: a pointer to an array of $t \times q$ configurations.

The index of a $t_i \times q_i$ path becomes the next parameter $t_{i+1}$, the relationship between two adjacent link paths is shown in Figure A.3.

The final full degree-of-freedom path can be easily retrieved back from the final $t_n \times q_n$ path to the first link path $t_1 \times q_1$ by recursively indexing with the $t$-coordinate. The full path is stored in an array of $DOF \times dim$. The structure for the full path is as below. Where $\text{dim}$ is the length of the final path.

```c
typedef struct dofpath {
    int dim;
    A_Float **d;
} DOFPath;
```
where,

- \( \text{dim} \): the dimension of the full-DOF path.
- \( d \): a pointer to an array of configurations.
References


REFERENCES


REFERENCES


REFERENCES


