GEOMETRICAL MOTION PLANNING FOR HIGHLY REDUNDANT MANIPULATORS USING A CONTINUOUS MODEL

APPROVED BY

SUPERVISORY COMMITTEE:

_____

_____

_____

_____

_____

To my family: Sumiko, Ayako, and Yu

GEOMETRICAL MOTION PLANNING FOR HIGHLY REDUNDANT MANIPULATORS USING A CONTINUOUS MODEL

by

AKIRA HAYASHI, B.S.,M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN
August, 1994

Acknowledgments

I would like to extend my deepest gratitude to my advisor Dr. Benjamin J. Kuipers for his advice, encouragements, and supports. Without his help, this doctoral research could not have been completed.

I am also grateful to Dr. Donald S. Fussell, Dr. Robert A. van de Geijn, Dr. Raymond J. Mooney, and Dr. Robert A. Freeman for their serving as the committee members.

Many thanks to my friends, Chris Walton and Richard Froom for taking on the tiresome work of proofreading this dissertation, and to David Throop for his development of POS: the PostScripting facility which was used to include many of the figures in this dissertation.

AKIRA HAYASHI

The University of Texas at Austin

August, 1994

GEOMETRICAL MOTION PLANNING FOR HIGHLY REDUNDANT MANIPULATORS USING A CONTINUOUS MODEL

Publication No. _____

Akira Hayashi, Ph.D.
The University of Texas at Austin, 1991

Supervising Professor: Benjamin J. Kuipers

There is a need for highly redundant manipulators to work in complex, cluttered environments. Our goal is to plan paths for such manipulators efficiently.

The path planning problem has been shown to be $PSPACE$-complete in terms of the number of degrees of freedom (DOF) of the manipulator. We present a method which overcomes the complexity with a strong heuristic: utilizing redundancy by means of a continuous manipulator model. The continuous model allows us to change the complexity of the problem from a function of both the DOF of the manipulator (believed to be exponential) and the complexity of the environment (polynomial), to a polynomial function of the complexity of the environment only.

The power of the continuous model comes from the ability to decompose the manipulator into segments, with the number, size, and boundaries of the segments varying smoothly and dynamically. First, we develop motion schemas for the individual segments to achieve a basic set of goals in open and cluttered space. Second, we plan a smooth trajectory through free space for a *point robot* with a maximum curvature constraint. Third, the path generates a set of position subgoals for the continuous manipulator which are achieved by the basic motion schemas. Fourth, the mapping from the continuous model to an available jointed arm provides the curvature bound and obstacle envelopes required (in step 2) to guarantee a collision-free path.

The validity of the continuous model approach is also supported by an extensive simulation which we performed. While the simulation has been performed in 2-D, we show a natural extension to 3-D for each technique we have implemented for the 2-D simulation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the first section, we introduce the motivation for the thesis subject: geometrical motion planning for highly redundant manipulators. Section 1.2 reviews research on highly redundant manipulators in general. Section 1.3 shows that current path planning methods cannot take advantage of the redundancy of such manipulators. Generally, redundancy is utilized only in the control level, after planning has been finished (Section 1.4). Our approach, geometrical motion planning using a continuous manipulator model, utilizes redundancy from the initial stage of motion planning. It is illustrated using a swan's neck scenario in Section 1.5. The overview of our approach follows in Section 1.6, where the thesis organization is also explained. Related research will be introduced in the last section.

1.1 Motivation

"What can be efficiently automated?" is a fundamental question in the discipline of computer science [Arden 80], and human's (or animal's) problem solving capabilities give us a weak existence proof for a class of problems which can be efficiently automated. The following paragraph on computer vision research [Barrow and Tenenbaum 78] expresses this kind of motivation.

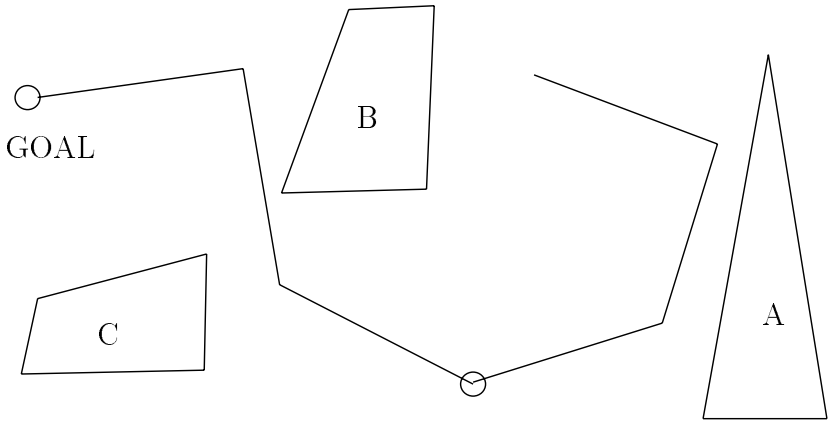


Figure 1.1: Path Planning Problem: finding a collision free trajectory to reach the goal. A,B, and C are obstacles.

> Attempts to construct computer models for the interpretation of arbitrary scenes have resulted in such poor performance, limited range of abilities, and inflexibility that, were it not for the human existence proof, we might have been tempted long ago to conclude that high performance, general purpose vision is impossible.

A similar statement can be made about path planning for manipulators, which this thesis is about. The path planning problem is the problem of finding a collision free trajectory for a manipulator between an initial state and a final state to reach a goal, when its environment is known (Fig. 1.1). Previous algorithms for path planning are either computationally expensive or unreliable, although humans seem to be good at path planning with their arms. Think of the task of reshelving books in a library, or of building toy castles with blocks. In these tasks, the working space is cluttered and each time we move an object (a book or a block), the layout of obstacles changes, which should make the problem even more difficult.

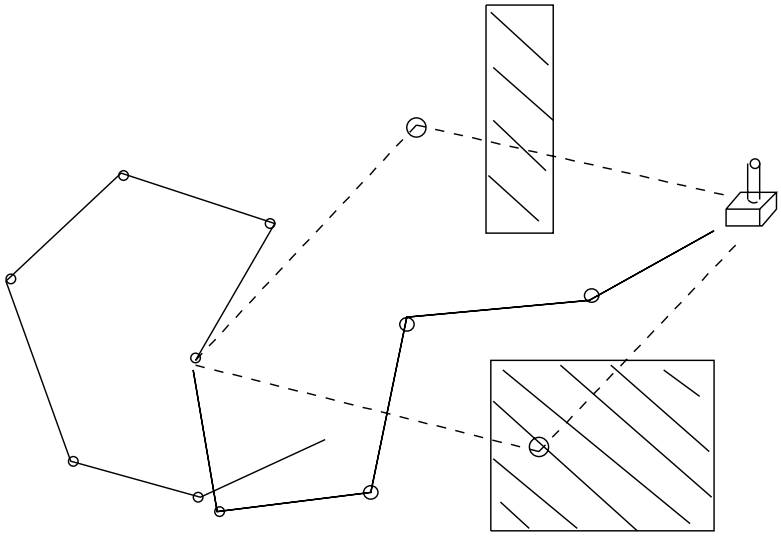


Figure 1.2: Redundancy Helps to Avoid Obstacles. Dotted lines represent a non-redundant manipulator which is unable to reach the goal.

We believe that this human performance to reach in cluttered space efficiently is attributed largely to the kinematic redundancy of our arm-body system. Redundant manipulators have more degrees of freedom (DOF) than necessary for a specified class of tasks. In order to reach a point in 3-D space, three DOF are necessary. In order to reach a point in space with a specified orientation, six DOF are necessary. Extra degrees of freedom can be used to perform other tasks such as singularity avoidance and minimization of joint displacement. In Fig. 1.2, the extra degrees of freedom is used to reach a point while avoiding obstacles. See [Hemami 90] for a list of such tasks.

Our arm itself has seven DOF, three at the shoulder, two at the elbow, and two at the wrist, and hence is redundant. We can reconfigure our arms without changing the hand position and orientation. Furthermore, we can obtain additional sources of redundancy by moving and/or twisting our body, which may be counted as six more DOF. Whenever we reach something, we (unconsciously) position our shoulder so that we can reach it more easily. This

observation leads us to a conjecture that path planning for redundant manipulators can be efficiently automated, while path planning for non-redundant manipulators remains difficult.

Although our main interest in this dissertation is in highly redundant snake-like manipulators rather than ones like human arms, we hope the results obtained for highly redundant manipulators will shed light on the path planning process done by humans for their arms.

1.2 Highly Redundant Manipulators

Many animals have highly redundant bodies and appendages, and it is evident that research on highly redundant manipulators and continuous arms in the robotics community has been motivated by animal morphology. [Wilson 83] studied the animal structures and motions of jumping spiders, nemerteans, and squid from the viewpoint of robotic design. [Horn 75] developed the kinematics, statics and dynamics of an eel-like locomotion system. [Hirose and Morishima 90] says

> One of the authors (Hirose) first began investigating the importance of the shape of the articulated body by studying the biomechanics of snakes. The study, begun in 1971, started from a spontaneous curiosity as to why snakes, having no legs, can move as smoothly as a flow of water over all kinds of terrain.

Various aspects of designing and controlling highly redundant artificial manipulators have been studied. The manipulators have been given a variety of names including ORM (the Norwegian word for snakes) [Pieper 68], spine robot [Drozda 84, Todd 86], snake-like manipulator [Clement and Iñigo 90],

elastic manipulator [Hirose *et al.* 83], elephant's trunk like elastic manipulator [Morecki *et al.* 87], and tentacle manipulator [Ivanescu and Badea 84]. Some were actually built [Pieper 68, Todd 86, Hirose *et al.* 83, Ikuta *et al.* 88, Hirose and Morishima 90]. While many of them are so called continuous arms, highly articulated arms [Lowe 85, Clement and Iñigo 90] and snake-like mobile robots [Waldron *et al.* 87, Hirose and Morishima 90] have also been studied.

Their applications are hard-to-reach painting jobs [Drozda 84], passing through restricted passages such as manholes for investigation and repair in contaminated area [Hemami 85], and active endoscope [Ikuta *et al.* 88]. These applications are aimed at utilizing the dexterity of highly redundant manipulators to work in complex, cluttered environments. In addition to these applications, [Pettinato and Stephanou 89] proposed to use a highly redundant manipulator as an all-in-one arm and gripping device capable of a wide variety of configurations and grasps.

Weight is the central problem in the mechanical design of the highly redundant manipulators because such manipulators consist of many elements, each of which needs an actuator and a sensor [Hemami 85]. Use of various actuators has been proposed including shape memory alloys [Ikuta *et al.* 88, Ikuta 90] and magnetics [Shahinpoor *et al.* 86]. Actuators using shape memory alloys have the advantage of high force output for the size of the actuator [Andeen 88]. Magnetical actuation has the same advantage. Different architectures for the mechanical structure of continuous arms as well as different actuation systems are discussed in [Kokkinis and Wilson 88]. A new hollow link design for a tendon actuated manipulator is also investigated in the paper.

Putting actuators at the remote base and controlling each element through tendons is a popular approach to the weight problem. Of such ma-

nipulators, the most famous is probably the Swedish built *Spine* robot, named after its similarity to human spines [Drozda 84, Todd 86]. It was not only studied, but was also built and put on the market. It is made up of a large number of disks put together by cables, and is actuated by pulling one or more cables. Although the robot got lots of attention from prospective buyers, the business has failed because of mechanical problems associated with the tendon technology.

A tensor actuated elastic manipulator was built [Hirose *et al.* 83]. The manipulator is made up of a series of eight coil spring elements, and each spring element is actuated by three cables. Therefore, the manipulator has 24 cables pulled by 24 motors located at the base. [Taylor *et al.* 83] and [Morecki *et al.* 87] have proposed similar architectures.

[Clement and Iñigo 90] attacked the weight problem of highly redundant manipulators in a very unique fashion. In their snake-like manipulator,

- The base is not fixed and translates in order to accommodate snake-like motion.
- Rotations of individual joints are restricted and the only allowable motion for the manipulator is follow-the-leader type. Joint i rotates at such a rate as to have θ_i assume the angle θ_{i+1} previously held by joint $i+1$ in the time it takes for the base to advance by one link length.

They proposed a mechanical integrator drive mechanism to realize the snake-like motion in a compact way. In this mechanism, power is simply transmitted from the base to the tip, making separate tendons or actuators unnecessary.

Although much work has been done on the study of mechanical designs for highly redundant manipulators, little attention has been paid to kine-

matics and path planning for such manipulators. Everyone acknowledges the dexterity of highly redundant manipulators. But it is believed that programming them is very difficult. In fact, as we will see in the next two sections, no current algorithm efficiently utilizes the dexterity of highly redundant manipulators for obstacle avoidance.

1.3 Limitations of the Current Path Planning Algorithms

Intuitively, one would hope that the difficulty of path planning would be a function of the complexity of the environment, and perhaps would become somewhat easier as the manipulator becomes more flexible and more redundant, i.e., acquires more degrees of freedom.

However, path planning algorithms based on the configuration space approach [Lozano-Pérez 83b] are intractable in terms of the number of DOF, and thus are not suited to path planning for highly redundant manipulators. Algorithms based on the artificial potential field approach [Khatib 86] are more computationally feasible, but have a drawback inherent in their use of local optimization techniques: the *local minima problem.*

Path planning is an important component of task level programming and has been studied extensively. In Chapter 2, we will investigate the current literature on path planning from our viewpoint, and claim that no current path planning method is suited to highly redundant manipulators. We need a radically different path planning approach in order to utilize redundancy.

1.4 Utilizing Redundancy for Obstacle Avoidance

Previous research on explicitly utilizing redundancy for obstacle avoidance has some characteristics in common with each other.

- Developed techniques primarily deal with low degrees of redundancy (e.g. a 7 DOF manipulator with one redundant DOF).
- An end effector trajectory is given initially. Obstacle avoidance is done by trying to choose a sub-optimal configuration (from infinitely many) to avoid obstacles while following the end effector trajectory.
- How to obtain an optimization criterion for choosing a configuration is not made explicit.

[Yoshikawa 84] decomposes the task into two subtasks. The first subtask is to follow an end effector trajectory, and the second subtask is to come close to a taught arm posture as much as possible. The arm posture should approximate the configuration which avoids obstacles. In [Kirćanski and Vukobratović 86], instead of teaching a desirable arm posture to avoid obstacles, the minimum distance between the manipulator and obstacles and the point on the manipulator nearest to obstacles are assumed to be obtained by the sensor system. If the distance goes below some safety margin, the motions of the links up to the link containing the nearest point are braked. [Maciejewski and Klein 85] took a similar approach. It was assumed that the nearest point and its desirable velocity which directs the point away from the obstacle surface can be obtained as a function of time.

In these three lines of research, the end effector trajectory is given. In [Jacak 89], the use of existing path planning methods for a *point robot* to

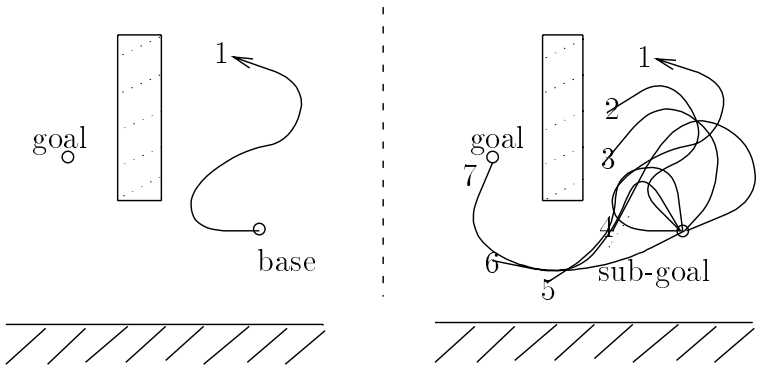


Figure 1.3: Scenario for Achieving Goal Position

obtain an end effector trajectory is proposed. If a collision appears for all configurations which follow the desired end effector trajectory, the end effector trajectory must be modified somehow. However, he did not suggest a solution to this problem.

While these methods are mainly for controling a robot manipulator after the planning of an end effector trajectory is finished, we here propose a new approach to the problem in which redundancy is utilized more aggressively from the first stage of path planning.

1.5 Swan's Neck Scenario for Path Planning

An example may be helpful to explain our main ideas. Consider a swan's neck as an example of a highly redundant manipulator (Fig. 1.3). Starting at position 1, the swan needs to reach the goal shown in the figure. First, the swan's neck must pass through the channel below the obstacle, so it sets a subgoal at the beginning of the channel. Second, it moves smoothly within an unobstructed convex region, to reach the subgoal at position 4. Third, redundancy becomes critical, as the swan simultaneously extends a progressively longer segment of its neck toward the goal, while using a progressively

shorter segment to maintain the subgoal position. Finally, at position 7, the swan reaches its goal while passing through the subgoal.

1.6 Overview of Our Approach

We simulate the above scenario by exploring path planning for a *continuous manipulator*. Figure 1.4 is an overview of a motion planning system which we have developed.

In Chapter 2, we review the current path planning algorithms with respect to their applicability to highly redundant manipulators. The results of the chapter suggest the need for a new and completely different approach. Chapter 3 describes the structure and control of the continuous manipulator model which we use to model highly redundant manipulators. It is controlled not by joint angles but by the continuously-changing curvature and torsion functions along the length of the manipulator. Once a path has been found for the continuous manipulator, the solution may be mapped back onto a jointed arm (Figure 1.5).

In Chapter 4, open-space routines which solves a planning problem within an unobstructed space, using a segment of the manipulator are described. In Chapter 5, we develop a set of motion schemas adequate for solving each of the basic problems that arise when the continuous manipulator moves through an obstructed environment. The approach employs decomposition which exploits the continuity of the model and the fact that the segment boundaries can be moved smoothly along the length of the curve to allow multiple subgoals to be achieved or maintained simultaneously.

When the space is obstructed, we search free space for a set of such subgoals. We reduce the problem of finding suitable subgoals for the continuous

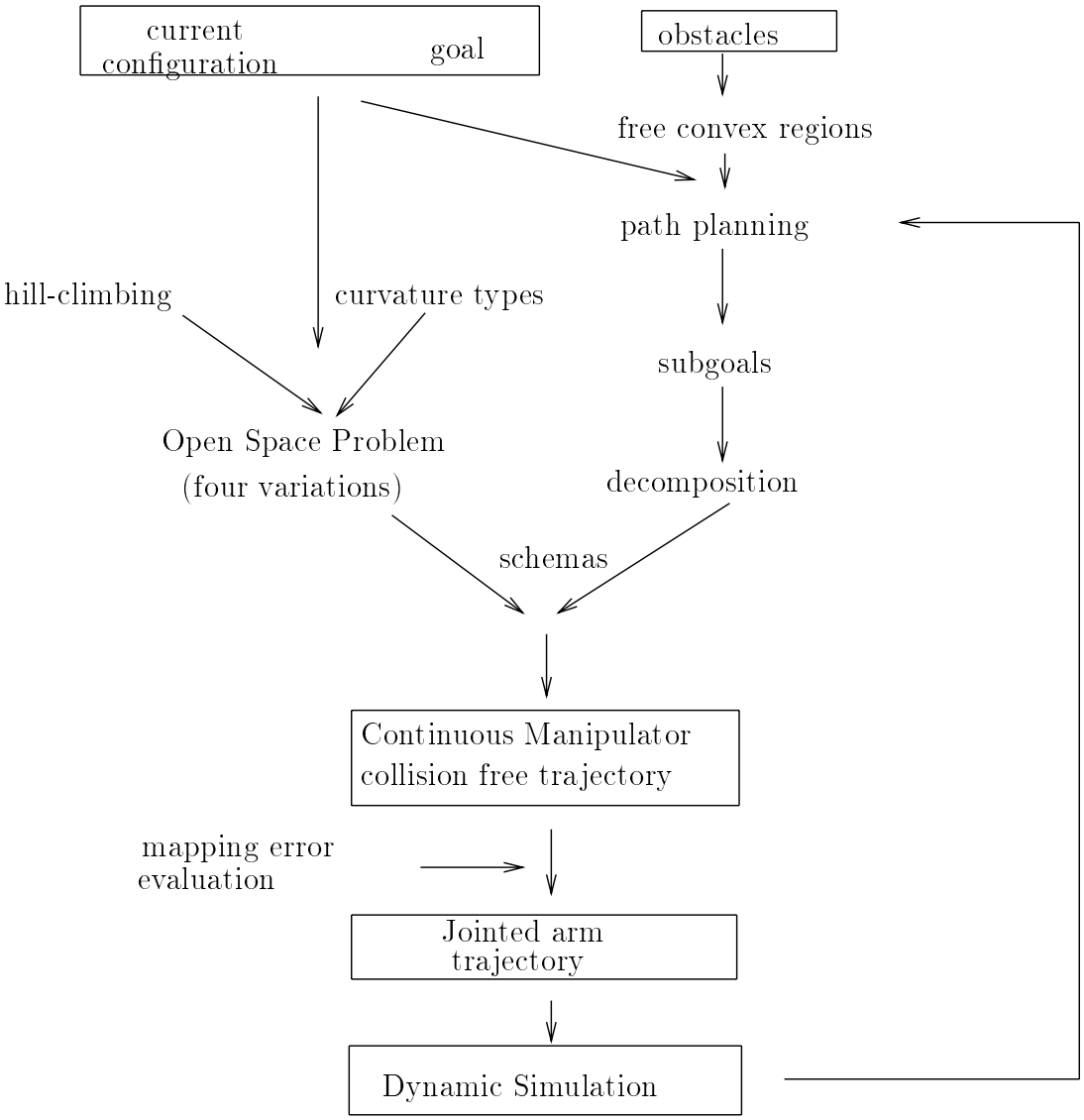


Figure 1.4: Overview of the Motion Planning System

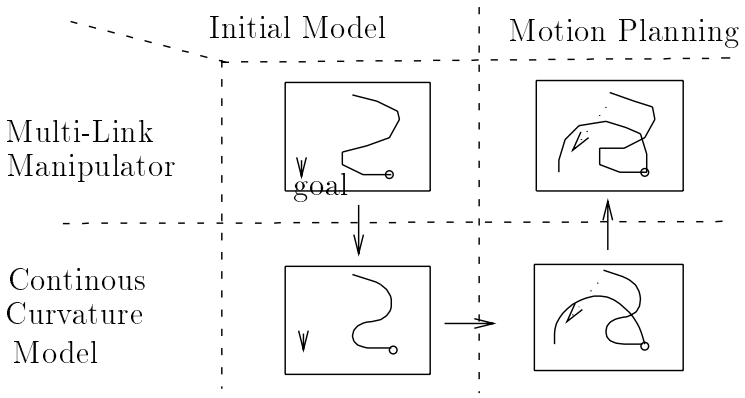


Figure 1.5: Solution Sequence of Our Approach

manipulator to the problem of finding a smooth, maximum-curvature path for a mobile robot in the same space, which we solve in Chapter 6. By combining the result of previous two chapters, we can obtain a collision free trajectory for the continuous manipulator in Chapter 7. By constraining the maximum curvature of a path and growing obstacles by an appropriate amount, we can guarantee that the mapping back to the original jointed arm is also collision free. Chapter 8 defines a maximum-curvature bound adequate to ensure collision-free motion as a function of the jointed arm.

The validity of the continuous model approach is also supported by an extensive simulation which we performed. While the simulation has been performed in 2-D, it will be shown in relevant chapters that there is a natural extension to 3-D for each technique we have developed. In particular, an extension from the continuous curvature model in 2-D to the continuous model in 3-D with curvature and torsion will be explained in Section 3.4. An extension of 2-D path planning to 3-D will be explained in Section 7.3. We plan to build a 3-D simulator as discussed in Section 9.3.1.

The major advantage of the continuous approach is that the difficulty of path planning decreases with the number of DOF in the jointed arm. This is

because the maximum curvature allowed for a path increases with the number of DOF in the jointed arm.

1.7 Related Work

[Chirikjian and Burdick 90] presents an approach similar to ours. However, there are important differences between the two approaches. While we use 5 point interpolation to discretize curvature and torsion, they use a modal decomposition. In particular, they have derived a closed form solution to the inverse kinematics problem in 2-D for a particular class of curvature functions.

We believe our approach is more general in the class of curvature functions considered, and we expect this generality to be important when these methods are extended to closed-loop control. Furthermore, obstacle avoidance was accomplished by manual decomposition and selection of curvature functions in [Chirikjian and Burdick 90]. Also, the problem of bounding the error in the mapping from continuous model to jointed arm is not addressed.

In spite of these differences in matters of technical detail, we are greatly encouraged that highly redundant manipulators, previously thought to be uncontrollable, are receiving increasing amounts of attention by high quality researchers.

Chapter 2

Current Approaches to Path Planning

In this chapter, we will review the current literature on path planning. First, the concept of task level programming is introduced, of which path planning is a component. Second, theoretical studies of the complexity of the path planning problem are reviewed. Third, many path planning algorithms are classified and reviewed. They are classified into four approaches to path planning: the configuration space approach, the heuristic configuration space approach, the potential field approach, and the hybrid approach. They are discussed in terms of their applicability to highly redundant manipulators. On the basis of these discussions, we claim that no current path planning algorithm is suited for highly redundant manipulators.

2.1 Task Level Programming

In [Lozano-Pérez 87b], the methods for robot programming were classified into 3 levels:

- Record and Playback
- Explicit Programming
- Task Level Programming

The most common method of robot programming is record-and-playback. In this method, humans teach the robot by manually moving the robot or using

a *teach pendant*, a hand-held button box which allows control of each DOF of a manipulator. The movement is recorded and then the robot repeats the sequence of motions exactly as it was taught. This rigid method of controlling a robot limits its tasks to those where there is very little interaction between itself and its environment. Welding and painting in factories are such tasks.

The apparent limitations of the record-and-playback method gave rise to another method, the explicit programming method. In this method, the movement of robots is specified by a program. High level programming languages with an extension for controlling robots are provided for this purpose. For example, AML [Taylor *et al.* 82], developed by IBM, has the following subroutines as an extension.

- PMOVE/DPMOVE: move manipulator to the absolute/relative coordinate
- LINEAR: move in linear motion
- PAYLOAD: set speed
- GRASP/RELEASE: close/open gripper
- DELAY: delay next command execution for a specified period
- TESTI/WAITI: test DI (digital input) point for value and branch, and wait for DI point to reach value (for synchronizing sub-tasks)

The advantage of this method is that a robot can interact with its environment through sensing, which enables the robot to cope with uncertainties. Output from sensing can be sent to a digital input point, which is read by

TESTI/WAITI subroutines. We can change the coordinates specified in the PMOVE/DPMOVE subroutines according to the readings.

Although this is an improvement over the record and playback method, the explicit programming of robotic applications is not easy. How to grasp a part depends on the shape, size, and location of both the part itself and other nearby objects. It is also dependent on how to approach the part. In general, conditions for performing each action are geometrical and dependent on a particular environment. Geometry is a domain where programming is known to be notoriously difficult, one reason why computational geometry is gaining more attention recently. Uncertainty in sensing and actions inherent to robotic applications make the programming even harder. It is difficult to develop such programs from scratch without any high level interfaces.

The goal of task level programming is to raise the level of robot programming from the detailed considerations of specific geometries and motions to the level of strategies. Rather than specifying "MOVE (100,100,10), ... GRASP", we want to be able to merely specify "PICK-UP Part-A". Task level programming is a very active topic in research. The following is a set of problems that must be solved to realize task level planning [Lozano-Pérez 87b].

- Path Planning
- On-Line Obstacle Avoidance
- Grasp Planning
- Fine-Motion Planning
- Uncertainty Modeling

Path Planning, along with on-line obstacle avoidance, is an important component of task level planning which can extend the application of robots from simple welding and painting tasks to more difficult assembly tasks.

2.2 Complexity of Path Planning Problems

As far as the path planning of highly redundant manipulators is concerned, the most closely related theoretical study is the study of the complexity of so called the generalized piano movers' problem. In the generalized piano movers' problem, a robot system itself (its encoding) is also an input to the problem, and a general algorithm is sought to find a collision free path for any robot system, instead of finding one for a particular robot system.

The problem has turned out to be extremely difficult. The problem has been shown to be $PSPACE$-complete[1] in terms of the total number of degrees of freedom (DOF) of the robot system. It is believed that even the best algorithm for the problem runs in exponential time for this class of problems. [Reif 79] showed PSPACE-hardness as a lower bound for the complexity of the problem. [Schwartz and Sharir 83] presents a double exponential algorithm. Later, Canny developed a single exponential algorithm called the road map algorithm [Canny 88a]. Furthermore, Canny showed the road map algorithm can be programmed to run in $PSPACE$ [Canny 88b], which gives its upper complexity bound: $PSPACE$-easiness.

If we fix the robot system (fix DOF), we get an algorithm which runs in polynomial time in the complexity of the environment. However, the algorithm has a huge constant factor.

[1] $LOGSPACE \subseteq PTIME \subseteq NPTIME \subseteq PSPACE$

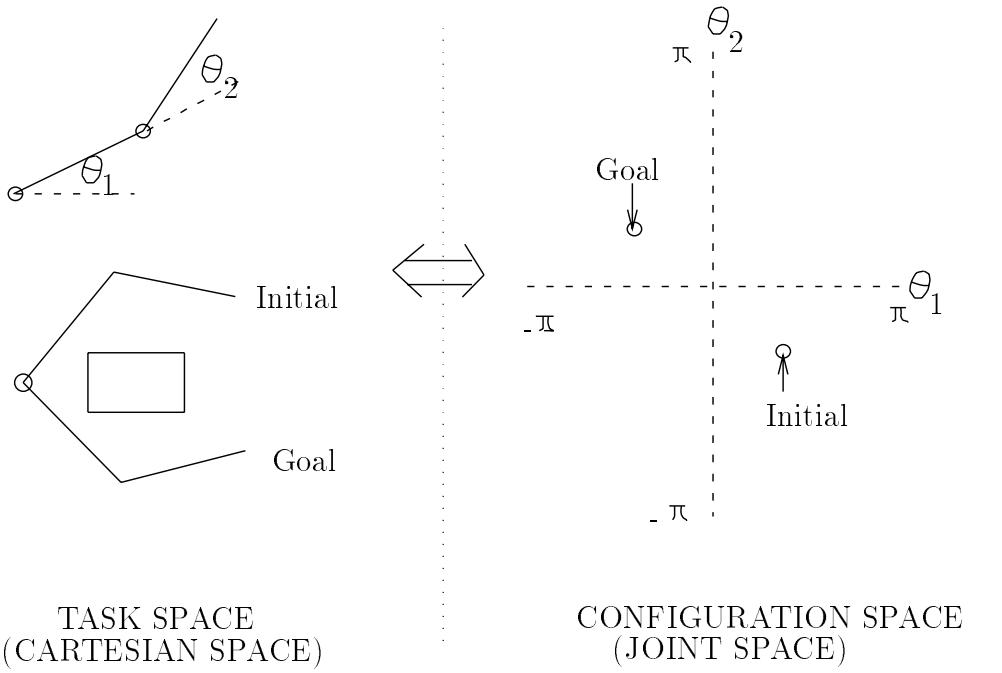


Figure 2.1: Task Space and Configuration Space

2.3 Configuration Space Approach

The configuration space approach was originally developed by Lozano-Pérez [Lozano-Pérez 83b]. In this approach, the path planning problem is not solved in the original two or three dimensional task space, but in the so called *configuration space*: the space of joint variables. In configuration space, a configuration of a manipulator is represented as a point (see Fig. 2.1). By mapping the obstacles from the task space to the configuration space (mapped obstacles are called *configuration space obstacles*), the path planning problem for a manipulator becomes a problem for a point robot in high dimensional space whose dimension is the number of DOF of the manipulator. For example, a problem in Fig. 2.1 is transformed to a problem in the configuration space by computing its configuration space obstacles (Fig. 2.2).

The algorithms based on the configuration space approach generally

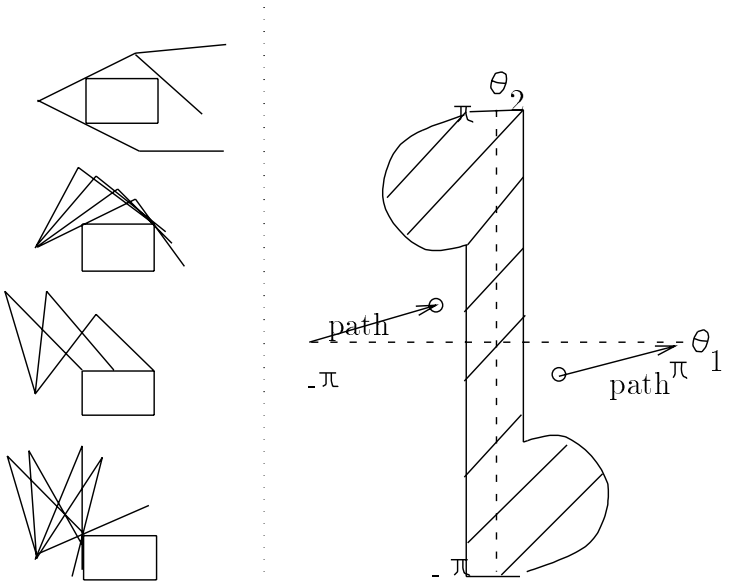


Figure 2.2: Obstacles in Configuration Space: manipulator configurations in the left generate the boundary of the configuration space obstacle.

proceed as follows.

1. Represent free space in the configuration space
2. Divide free space into cells
3. Build an adjacency graph between cells
4. Find the cells for the initial and the final configuration
5. Find a path in the adjacency graph using the A^* search

The dominant factor in complexity is *representing* free space in the configuration space, the first step in the above procedure.

In [Lozano-Pérez 87a], the free space map is built recursively by finding the legal ranges of link joints from the base link (link 1) up to the most distant link (link n) as follows.

1. Find the ranges of legal values for q_i (the rotation of joint i, the joint between link $i-1$ and i). The range is obtained by computing the area link i sweeps when the link rotates around the positions of the joint i determined by the current value ranges of $q_1, \ldots, q_{i-1}$.

2. Sample the legal range of q_i at some resolution (e.g. 2°), and compute the legal range for the next joint q_{i+1}.

Thus, the legal ranges of joints make a tree structure.

Lozano-Pérez's algorithm can improve the worst case complexity of exact algorithms for the generalized piano movers' problem[2] by controlling the resolution. The algorithm is complete, because the algorithm is guaranteed to find a path if it exists by increasing the resolution. However, its worst case complexity is $O(r^{k-1}(mn)^2)$, when the manipulator has k DOF, the joint ranges are divided into r intervals, the manipulator is described with m faces and edges, and the obstacles are described with n faces and edges. The complexity is exponential in terms of the number of DOF. It has been said that the approach is impractical for more than 4 DOF [Faverjon and Tournassoud 89].

2.4 Heuristic Configuration Space Approach

Because of the complexity of the algorithms associated with the configuration space approach, various heuristics have been proposed within the framework to make path planning more efficient. However, none of them seem to be applicable to highly redundant manipulators.

A common practice is to decompose a manipulator in two parts: its

[2]Exact algorithms such as Canny's road map algorithm were never implemented.

arm and its hand. Gross motion is planned for its arm with occasional reorientation of the hand [Hasegawa and Terasaki 88, Lozano-Pérez *et al.* 90]. But these techniques apply only to 6 DOF non redundant manipulators.

[Dupont and Derby 86] attacked the problem of applying the configuration space approach to redundant manipulators. Their idea is to build as little as possible of a configuration space map to find a path. They do not build a complete configuration space map (at some resolution) at the beginning, because this map building is the most costly step. The configuration space is recursively subdivided into regions (hypercubes) which is represented as a tree structure. Their algorithm uses the following heuristics to guide a search for a path through the regions.

- Try to minimize the path length in the configuration space.
- Favor large regions of free space.

Only when the search reaches a region in the tree is the mapping from the task space to the configuration space done and is it determined whether the region is free, occupied, or partially free.

We think it is difficult to apply their approach to highly redundant manipulators because the complexity of the mapping is exponential, no matter whether it is part or whole of the configuration space map. Moreover, their heuristics are too weak. They make no use of the idea of utilizing redundancy for obstacle avoidance. When their heuristics do not work, they eventually have to build most of the map.

[Gupta 90] proposes an interesting approach. He uses a strong heuristic to eliminate the necessity to build a high dimensional configuration space

map. His basic idea is to plan the motion of each link sequentially, starting from the base link. Suppose the motion of the links up to link i has been finished. This gives us the trajectory of the joint between link i and $i+1$. By neglecting the distant links $(i+2, \ldots, n)$, planning the motion for link $i+1$ becomes a two dimensional path planning problem, a problem of finding a path for a single link whose one end follows a given trajectory. While eliminating the necessity of building a high dimensional configuration space map is attractive, his approach has some problems.

- He assumes that a goal configuration of final joint positions is given. However, for redundant manipulators there are infinitely many configurations which reach a goal point.

- His heuristic is not well suited for highly redundant manipulators. One of the typical motions for a snake-like robot in [Clement and Iñigo 90] is to follow a trajectory of the tip joint by the succeeding joints. The motion is planned sequentially, but starting from the tip link. This is in reverse order to Gupta's planning sequence.

2.5 Artificial Potential Field Approach

Another popular approach for path planning is called the artificial potential field approach. It was originally proposed by Khatib [Khatib 86]. The approach is based upon creating an artificial potential field in the task space. The potential field is generated by adding an attractive force toward the goal point and repulsive forces from obstacles. The manipulator is supposed to reach the goal point while avoiding the obstacles by simply following the steepest decent of the potential field (see Fig. 2.3). This approach is more

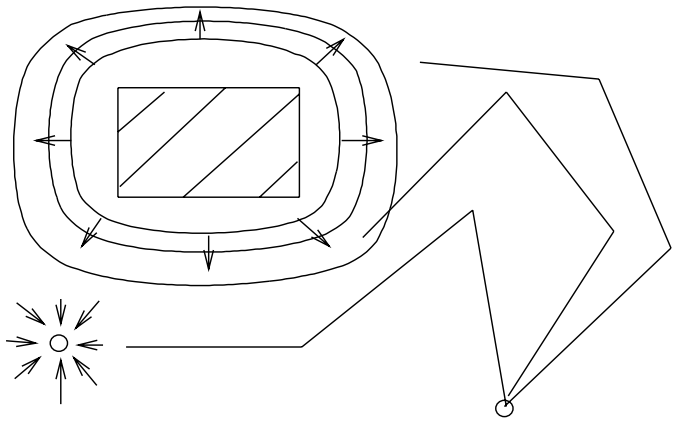


Figure 2.3: Artificial Potential Field

computationally feasible than the configuration space approach, because the planning is carried out in the original task space and no mapping to the configuration space is done. Another advantage is that the approach also addresses the control problem, since the artificial potential field naturally induces a feedback control law.

However, the approach has a severe drawback inherent in its use of a local optimization technique: *the local minima problem* (Fig. 2.4). While the potential field approach can be applied to path planning for both point robots and manipulators, the local minima problem is more serious for manipulators. Note that local minima can appear for manipulators in the same environment in which there is none for point robots. This is because points subject to repulsive forces must be put not only at the tip but also at many places along a manipulator in order for all parts of the manipulator to avoid obstacles.

In an attempt to build a repulsive potential field around an obstacle without generating a certain class of local minima, [Khosla and Volpe 88] pro-

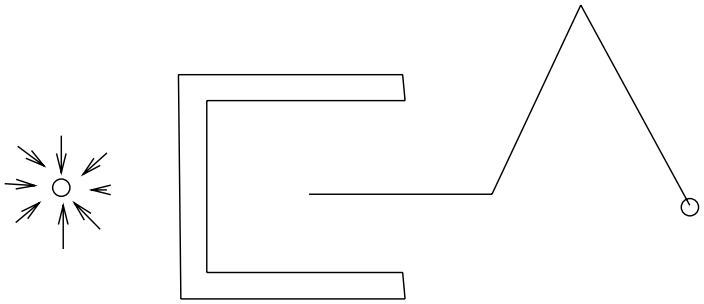


Figure 2.4: Getting Caught in a Local Minimum

posed to use superquadratic artificial potential functions in lieu of the FIRAS[3] potential functions proposed by Khatib. A Superquadratic function is chosen because its contour of the potential change from the object shape near an object, to a spherical shape away from the object. When a repulsive potential field which has spherical contour lines is added to an attractive potential field which also has spherical contour lines, local minima are not created. Using superquadratic functions improves the situation. But it does not eliminate the problem. Local minima still exist near obstacles where contour lines of the potential field has non-spherical shapes.

[Rimon and Koditschek 89] tries to construct *navigation functions*. A navigation function is an artificial potential energy function on a robot's configuration space which has no local minima. They succeeded in constructing one for a limited class of environments which they call a sphere worlds and a star worlds. A sphere world is an n-dimensional disc in E^n punctured by a finite number of smaller disjoint n-dimensional discs. Smaller discs represent obstacles. But as we can imagine from the theoretical study, the complexity of building these navigation functions increases exponentially as a function of the number of DOF of a manipulator. Their approach may have an interesting

[3]from the French, Force Indusing an Artificial Repulsion from the Surface

application in the feedback control of motions, but it has a limited impact on path planning.

[Barraquand and Latombe 89] presents another way to deal with local minima. It is called a Monte-Carlo approach. Instead of eliminating local minima, the authors try to get out of a local minimum by giving the manipulator several random motions, each of these motions having a random duration. From all the terminating configurations after the random motions, they follow the gradient of the potential field and reach new local minima again. Only those minima which have the lower potential than the previous local minimum are retained, and this exit procedure continues in a best first search fashion until they get to a goal (or the dead end).

Redundancy has both positive and negative effects for the approach. The number of solutions (paths) increases with the number of DOF in the manipulator, and it may make the best first search efficient. However, the number of local minima also seems to increase. While the results of their experiments are impressive (they showed a path for an 8 DOF serial manipulator), it remains to be studied up to how many DOF their approach is applicable.

2.6 Hybrid Approaches

As we have seen, the configuration space approach is classified as a global planning approach, while the potential field approach is classified as a local planning approach. In the hybrid approach, both global and local approaches are combined. The approach was taken in the mobile robot domain [Krogh and Thorpe 86], where a global planner finds subgoals which are then traced by a local maneuver. The difficulty of this approach lies in distinguishing the global level from the local level.

[Warren 89] presents a hybrid approach in a manipulator domain. First, a configuration space map is built. Then, a path is found by putting an artificial potential field in the configuration space. This method tries to solve the biggest problem in the potential field approach, the local minima problem. Unfortunately, it also loses the biggest advantage of the potential field approach, its computational feasibility. Building a configuration space map is the most costly step.

The opportunistic global path planner in [Canny and Lin 90] does not build a configuration space map. In their approach, a robot moves along skeleton curves which are constructed incrementally until a path is found. The skeleton curves are the loci of maxima of an artificial potential function whose potential is proportional to the minimum distance between the robot and obstacles. In two dimensional space, the skeleton curves are those found in a Voronoi diagram. First, the nearest local maxima are found on the skeleton curves for both an initial and a goal configuration. Let us call these points of local maxima P_{init} and P_{goal}. Then, the skeleton curve with P_{init} is traced to search for P_{goal}. If it cannot find P_{goal} on the curve, the planner takes a slice projection through a critical point to explore another skeleton curve. The planner repeats the process until it finds P_{goal}. By incrementally building skeleton curves, the approach can improve the average case running time for path planning. By exploring all the skeleton curves eventually, the planner is guaranteed to find a path if one exists. However, this property also shows the algorithm is intractable. In fact, the total number of critical points which are useful in reaching goals is $O(n^{k-1})$, where n is the complexity of the environment and k is the number of DOF of the robot.

An approach proposed in [Faverjon and Tournassoud 89] is aimed at

finding paths for manipulators with many degrees of freedom. They do not build a configuration space map, because building one is too expensive for manipulators which have more than four DOF. They divide the whole configuration space into relatively large cells. No information on the obstacles is associated with the cells. Then a connectivity graph is built. A node of the graph is a cell in the configuration space. An arc of the graph represents the probability for a local planner to succeed to move from one cell to another without getting caught at a local minimum. The graph is searched for the most promising path between the two cells, one containing the initial configuration and the other containing the goal configuration. The center of the cells along the path are subgoals to be reached one after another by the local planner.

The difficulty of the approach lies in computing the probability for the arcs without building a configuration space map. The authors suggest initializing all the probabilities to the same value and updating them according to the results of motion trials along the arcs. This learning phase on a particular environment would take time. It seems that their approach is feasible only in very static environments.

2.7 Summary of the Chapter

The work which we have discussed in this chapter is relevant for manipulator path planning, but our problem, path planning utilizing redundancy, is not addressed. We have seen that no current path planning algorithm is suited for highly redundant manipulators.

Chapter 3

The Continuous Manipulator Model

This chapter describes the structure and control of the continuous manipulator model. It is controlled not by joint angles but by continuously-changing curvature and torsion, intrinsic properties of smooth curves, along the length of the manipulator.

By using the continuous model, we try to capture the macroscopic shape of a highly redundant manipulator. The shape of continuous arms along its center line can be directly expressed by the continuous model. Even for discrete (jointed) arms, their macroscopic shape can be expressed by the continuous model.

We briefly refer to rationales for a continuous model in Section 3.1. We then describe the continuous model in 2-D and in 3-D. Section 3.2 summarizes relevant topics from the differential geometry of plane curves. Section 3.3 explains the continuous model in 2-D, the continuous curvature model. In Section 3.4 and Section 3.5, the continuous model in 3-D is explained.

3.1 Why a Continuous Model?

There is little research on geometrical aspects of highly redundant manipulators. Probably, Pieper's research on a snake-like continuous arm (which is called the ORM manipulator) is the first on the kinematics of highly redundant manipulators [Pieper 68]. In one chapter of his dissertation, he attacked

the inverse kinematics problem for the ORM manipulator. In an attempt to control the shape of the continuous manipulator using a computer, he proposed to use a *digital manipulator* model. The 2-D model of the digital manipulator is made up of n links where the angle between two adjacent links can be either $+\theta_0$ or $-\theta_0$, where θ_0 is a constant.

To solve the inverse kinematics problem, he developed a search algorithm in the 2^n possible states. The search does not always find a solution because of local minima. Noticing that the search works more reliably when we start in a neighborhood of the solution, he proposed to find a coarse approximation of the solution by using four circular arc segments, with their radii and angles varied continuously. The circular approximation is then mapped back to the digital manipulator to give an initial state for the search.

We are going to discretize the continuous manipulator not by using a digital manipulator model, but by using a decomposition technique and an interpolation technique for continuous functions. As we will see, for each technique in literature developed for a discrete manipulator, an equivalent technique for the continuous manipulator model has been developed or found. In this way, without sacrificing the simplicity and easiness, we are able to exploit the advantages of using continuous functions, to which Pieper finally resorted in finding an approximate solution for the inverse kinematics problem. The power of the continuous model along with the decomposition technique is a main subject of the thesis and will be demonstrated in the following chapters.

3.2 Intrinsic Properties of Plane Curves

The continuous manipulator model we have developed is a mathematical model based on differential geometry of smooth curves. It is therefore

useful to summarize a few relevant properties of continuous curves without proofs. For a complete treatment on the subject, see any textbook on differential geometry (for example, [Stoker 69]). In this section, we explain plane curves.

3.2.1 Regular Curves

Here we deal with a class of plane curves called *regular curves*. A curve

$$\mathbf{P}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$
$$\alpha \leq t \leq \beta$$

is regular, if and only if the following conditions are satisfied.

1. $\mathbf{P}(t)$ has second continuous derivatives in the interval defined.
2. $\dot{\mathbf{P}}(t)$, called a tangent vector, is nowhere zero.

Note that if we change the parameter from t to τ by $t = \psi(\tau)$ such that

$$\dot{\psi}(\tau) \neq 0$$

then the resulting curve is also a regular curve.

3.2.2 Curve Length

For a regular curve, we can define the length of the curve L_α^β by the definite integral

$$L_\alpha^\beta = \int_\alpha^\beta \sqrt{\dot{\mathbf{P}}(t) \cdot \dot{\mathbf{P}}(t)} dt = \int_\alpha^\beta \sqrt{\dot{x}^2 + \dot{y}^2} dt$$

It is useful to consider the length, $s(t)$, from a fixed point to a variable point:

$$s(t) = L_\alpha^t = \int_\alpha^t \sqrt{\dot{\mathbf{P}}(\tau) \cdot \dot{\mathbf{P}}(\tau)} d\tau = \int_\alpha^t \sqrt{\dot{x}^2 + \dot{y}^2} d\tau$$

Since

$$\frac{ds(t)}{dt} = \sqrt{\dot{\mathbf{P}}(t) \cdot \dot{\mathbf{P}}(t)}$$

and

$$\dot{\mathbf{P}}(t) \neq \mathbf{0}$$

holds everywhere for a regular curve, we use the arc length s as a parameter of a curve. Then, a tangent vector becomes a unit vector:

$$|\dot{\mathbf{P}}(s)| = 1$$

The arc length of a curve is independent of the choice of a coordinate system. Moreover, it is invariant when we change a parameter t.

3.2.3 Curvature

The curvature of a plane curve is defined as follows. First, we define the orientation angle $\theta(s)$ such that it is a continuous function of s which satisfies

$$\tan\theta(s) = \frac{\dot{y}(s)}{\dot{x}(s)}$$

Then, a curvature function $\kappa(s)$ is defined as the derivative of $\theta(s)$:

$$\kappa(s) = \frac{d\theta(s)}{ds}$$

From the definition, curvature is invariant under a change of coordinate system which preserves the orientation of the axes. The formula for curvature using the components x and y is

$$\kappa = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \tag{3.1}$$

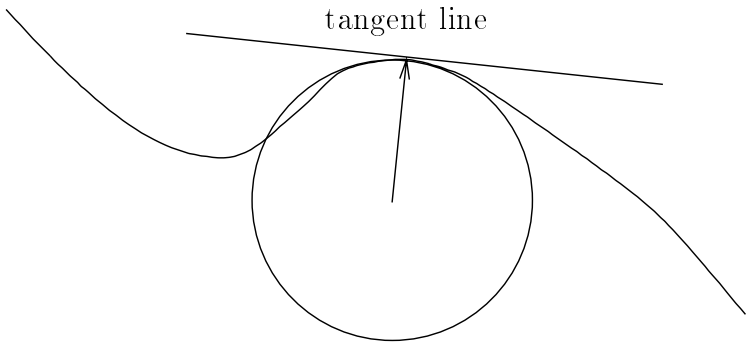


Figure 3.1: Osculating Circle and Radius of Curvature

At a given point on a curve, the circle that has the same first and second derivative vectors as the curve is called the osculating circle. Its radius is called the radius of curvature $\rho(s)$. The curvature $\kappa(s)$ at the point is the reciprocal, $\frac{1}{\rho(s)}$, of the radius of curvature (see Fig. 3.1).

3.2.4 Existence of a Plane Curve given Curvature

We have seen that arc length and curvature are invariant properties of regular plane curves. Now we introduce a theorem on the existence of a plane curve given the invariants.

Theorem 1 *For a given continuous function $\kappa(s)$ defined for $s_0 \leq s \leq s_1$, there is one and only one regular curve (within a rigid motion) such that $\kappa(s)$ is its curvature function and s is its arc length.*

The theorem can be proved by defining $\mathbf{P}(s)$ from $\kappa(s)$ as follows.

$$\begin{aligned} \theta(s) &= \theta(s_0) + \int_{s_0}^{s} \kappa(\sigma) d\sigma \\ \dot{\mathbf{P}}(s) &= \begin{pmatrix} \cos\theta(s) \\ \sin\theta(s) \end{pmatrix} \\ \mathbf{P}(s) &= \begin{pmatrix} x(s_0) + \int_{s_0}^{s} \cos\theta(\sigma) d\sigma \\ y(s_0) + \int_{s_0}^{s} \sin\theta(\sigma) d\sigma \end{pmatrix} \end{aligned} \tag{3.2}$$

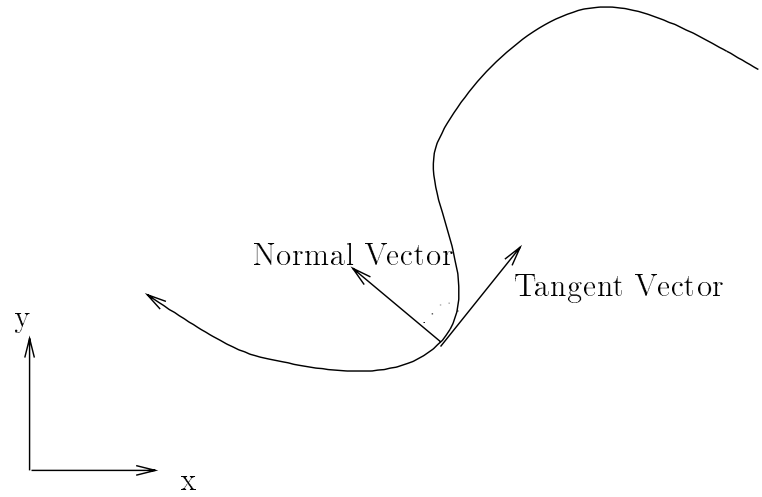


Figure 3.2: Tangent and Normal Vectors

From (3.2), it is shown that $\mathbf{P}(s)$ is a regular curve for which s is its arc length and $\kappa(s)$ is its curvature. It is also shown that any two curves $\mathbf{P}^1(s)$ and $\mathbf{P}^2(s)$ which have the same arc length and curvature differ at most by a rigid motion. □

3.2.5 Frenet Equations

Another way to establish the previous theorem is through the Frenet Equations. The Frenet Equations are expressed using a pair of orthogonal unit vectors $\mathbf{v}_1(s)$ and $\mathbf{v}_2(s)$. $\mathbf{v}_1(s)$ is the tangent vector of a curve:

$$\mathbf{v}_1(s) = \dot{\mathbf{P}}(s)$$

The normal vector, $\mathbf{v}_2(s)$, is defined such that it is a unit vector normal to the tangent vector and the vectors $\mathbf{v}_1(s)$ and $\mathbf{v}_2(s)$ have the same orientation as the coordinate axes. See Fig. 3.2.

Every vector is expressed as a linear combination of $\mathbf{v}_1(s)$ and $\mathbf{v}_2(s)$,

and it can be shown that vectors $\dot{\mathbf{v}}_1(s)$ and $\dot{\mathbf{v}}_2(s)$ are expressed as follows.

$$\begin{pmatrix} \dot{\mathbf{v}}_1(s) \\ \dot{\mathbf{v}}_2(s) \end{pmatrix} = \begin{pmatrix} 0 & \kappa(s) \\ -\kappa(s) & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1(s) \\ \mathbf{v}_2(s) \end{pmatrix} \tag{3.3}$$

The equations form a system of ordinary differential equations for $\mathbf{v}_1(s)$ and $\mathbf{v}_2(s)$, and are called the Frenet equations. We can also establish Theorem 1 through the Frenet equations by using the well known theorem on the existence and the uniqueness of a solution of ordinary differential equations.

3.3 Continuous Model in 2-D

The continuous model in 2-D is called the continuous curvature model. Its motion is controlled by its continuously-changing curvature function. Its configuration is represented by a series of curvature segments. The number of segments is controlled by a decomposition technique to dynamically control the degree of redundancy. Simple configurations can be represented by one segment. More complex configurations are represented by a series of segments.

3.3.1 Curvature Segment and Curvature Operators

A curvature segment is the basic unit of representation of the continuous model in 2-D. It is specified by length L, start position $(x(0), y(0))$, start orientation $(\dot{x}(0), \dot{y}(0))$, and a curvature function $\kappa(s)$. The curvature function is discretized using five points in the curvature graph: (s_a, κ_a), (s_b, κ_b), (s_c, κ_c), (s_d, κ_d), and (s_e, κ_e). Cubic spline interpolation is used to interpolate curvature from the five points[1]. To change the shape of the segment, curvature operators are defined to move the five points. See Figure 3.3.

[1] There is not a sufficient reason to use smooth cubic spline interpolation. Piecewise linear interpolation suffices for curvature to be continuous.

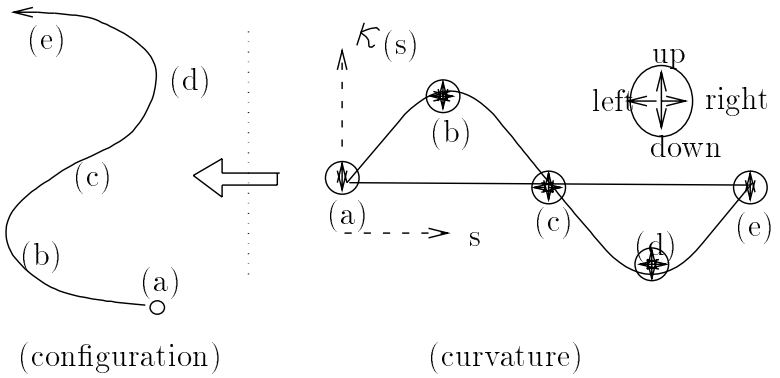


Figure 3.3: Curvature Segment Representation and its Operators. The following *curvature operators* are used to change curvature (and configuration). **a.** Increase/decrease κ_a, κ_b, κ_c, κ_d, or κ_e (move up/down an interpolation point). **b.** Increase/decrease s_b, s_c, or s_d. (move right/left an interpolation point.) **c.** Rotate the base of a curve.

A *configuration* of the curvature segment is a function $(x(s), y(s))$ for $0 \leq s \leq L$, which gives us the coordinates and the orientation of all points on the segment. The configuration is obtained numerically for a given curvature function $\kappa(s)$ with its initial condition: $(x(0), y(0))$ and $(\dot{x}(0), \dot{y}(0))$ (see Section 3.3.3).

3.3.2 Decomposition of Segment

The curvature segment representation alone is not rich enough to express complex configurations. Configurations with two or more inflection points[2] are necessary to achieve a goal while avoiding obstacles in cluttered space. The decomposition technique makes it possible to divide a segment into two or more segments (Fig. 3.4).

For a decomposition to be meaningful, we have the following decom-

[2] An inflection point is a point where the curvature function changes its sign.

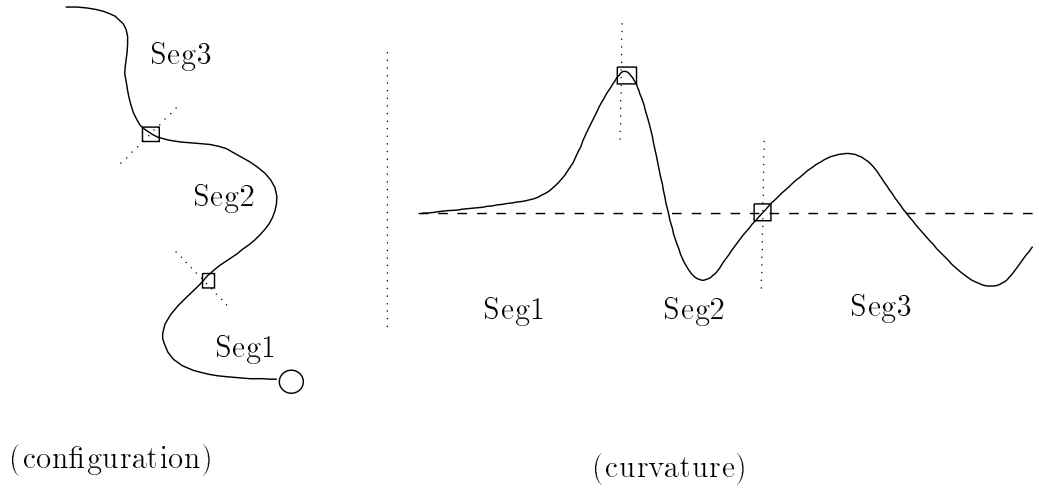


Figure 3.4: Decomposition of Segment

position rules;

- The total length of segments generated must be the same as that of the original segment.
- The orientation and the curvature must be continuous at a decomposition point[3].

These decomposition rules are the only constraint among segments, and each segment is controlled quite independently after decomposition.

Decomposition, as a general problem solving technique, has been applied to manipulator motion planning and control. It is a common practice in path planning to decompose a non-redundant 6 DOF manipulator into its arm and hand, each of which has 3 DOF. Gross motion planning can be done for the arm neglecting the hand if the size of the hand is much smaller than the that of the arm.

[3]A point where we divide a segment in decomposition is called a decomposition point.

[Lee and Lee 90] proposed a method to control a redundant manipulator by decomposing it into serially linked non-redundant manipulators. The problem of redundancy resolution is transformed into decomposing an end effector velocity given as a task into velocity component assigned to the tips of sub-manipulators called *task points*. Since each sub-manipulator has enough degrees of freedom, sub-tasks can be expressed in the same level as the original task given. Although they did not go into the subject of obstacle avoidance which is our main motivation for decomposition, their decomposition technique is similar to ours.

But there are still big differences between their decomposition technique and ours. Because of the continuity of the model, we have great flexibility in decompositions. In particular,

- We can use the same representation for the original segment and the decomposed segments.
- We can choose any point as a decomposition point.
- We can move a decomposition point smoothly along the length of the continuous model to make one segment longer while making the other shorter.

3.3.3 Obtaining a Configuration from Curvature

Here we show how to obtain a configuration from curvature. One way is by solving the curvature equation numerically.

$$\kappa = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \tag{3.4}$$

It will be convenient to simplify the equation by choosing an appropriate parameter. Let L be the total length of the segment. When we choose as the parameter $\tilde{s}$ which is the ratio of curve length s to the total length L [4] , $\tilde{s}$ satisfies the following.

$$L \cdot \tilde{s} = \int_0^{\tilde{s}} \sqrt{(\dot{x}(t))^2 + (\dot{y}(t))^2}\, dt$$

Differentiating the above by $\tilde{s}$, we get the next relation.

$$L = \sqrt{(\dot{x}(\tilde{s}))^2 + (\dot{y}(\tilde{s}))^2} \tag{3.5}$$

Then we substitute (3.5) to T(3.4) to rewirte the curvature equation.

$$\begin{aligned} \kappa(\tilde{s}) &= \frac{\dot{x}(\tilde{s})\ddot{y}(\tilde{s}) - \ddot{x}(\tilde{s})\dot{y}(\tilde{s})}{L^3} \\ 0 &\leq \tilde{s} \leq 1 \end{aligned} \tag{3.6}$$

Given curvature $\kappa(\tilde{s})(0 \leq \tilde{s} \leq 1)$ and boundary conditions $(x(0), y(0))$, $(\dot{x}(0), \dot{y}(0))$, we solve (3.6) numerically using the following algorithm.

initial: $\tilde{s} = 0$

set $\Delta\tilde{s}$ small enough for the iteration to converge

repeat: compute $(x(\tilde{s} + \Delta\tilde{s}), y(\tilde{s} + \Delta\tilde{s}))$ from $(x(\tilde{s}), y(\tilde{s}))$ and $(\dot{x}(\tilde{s}), \dot{y}(\tilde{s}))$

compute $(\dot{x}(\tilde{s} + \Delta\tilde{s}), \dot{y}(\tilde{s} + \Delta\tilde{s}))$ from $(\dot{x}(\tilde{s}), \dot{y}(\tilde{s}))$ and $\kappa(\tilde{s})$

set $\tilde{s} = \tilde{s} + \Delta\tilde{s}$

if $\tilde{s} < 1.0$ then goto repeat, else exit

[4] We chose to use the normalized length parameter $\tilde{s}$ rather than the actual length parameter s from some considerations for implementing the simulation program.

It is easy to compute $(x(\tilde{s}+\Delta\tilde{s}), y(\tilde{s}+\Delta\tilde{s}))$ from $(x(\tilde{s}), y(\tilde{s}))$ and $(\dot{x}(\tilde{s}), \dot{y}(\tilde{s}))$.

$$x(\tilde{s}+\Delta\tilde{s}) = x(\tilde{s}) + \Delta\tilde{s}\cdot\dot{x}(\tilde{s})$$
$$y(\tilde{s}+\Delta\tilde{s}) = y(\tilde{s}) + \Delta\tilde{s}\cdot\dot{y}(\tilde{s})$$

In order to get $(\dot{x}(\tilde{s}+\Delta\tilde{s}), \dot{y}(\tilde{s}+\Delta\tilde{s}))$, we approximate (3.6) by the following.

$$\dot{x}(\tilde{s})\cdot\frac{\dot{y}(\tilde{s}+\Delta\tilde{s})-\dot{y}(\tilde{s})}{\Delta\tilde{s}} - \dot{y}(\tilde{s})\cdot\frac{\dot{x}(\tilde{s}+\Delta\tilde{s})-\dot{x}(\tilde{s})}{\Delta\tilde{s}} = L^3\cdot\kappa(\tilde{s}) \tag{3.7}$$

We get the other equation for $(\dot{x}(\tilde{s}+\Delta\tilde{s})$ and $\dot{y}(\tilde{s}+\Delta\tilde{s}))$ from (3.5).

$$(\dot{x}(\tilde{s}+\Delta\tilde{s}))^2 + (\dot{y}(\tilde{s}+\Delta\tilde{s}))^2 = L^2 \tag{3.8}$$

By solving (3.7) and (3.8) simultaneously, we get $(\dot{x}(\tilde{s}+\Delta\tilde{s}), \dot{y}(\tilde{s}+\Delta\tilde{s}))$ from $(\dot{x}(\tilde{s}), \dot{y}(\tilde{s}))$. There are two algebraic solutions which satisfies (3.7) and (3.8), and the one which is closer to $(\dot{x}(\tilde{s}), \dot{y}(\tilde{s}))$ is what we want. The other one corresponds to the curve with the same curvature but with an opposite orientation.

Other ways to obtain a curve from curvature are by (3.2) and by the Frenet equations (3.3). The Frenet equations give us a straightforward expression for applying numerical integration to get curve shape from curvature (see also Section 3.5).

3.4 Intrinsic Properties of Space Curves

For many of the notions on plane curves explained in Section 3.2, natural extensions exist to space curves.

3.4.1 Regular Curve

A space curve

$$\mathbf{P}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$$
$$\alpha \leq t \leq \beta$$

is regular, if and only if the following conditions are satisfied.

1. $\mathbf{P}(t)$ has continuous third derivatives in the interval defined.
2. $\dot{\mathbf{P}}(t)$, called a tangent vector, is nowhere zero.

3.4.2 Curve Length

The length of curve $s(t)$ from a fixed point to a variable point for a space curves is

$$s(t) = L_\alpha^t = \int_\alpha^t \sqrt{\dot{\mathbf{P}}(\tau) \cdot \dot{\mathbf{P}}(\tau)} d\tau = \int_\alpha^t \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} d\tau$$

When we use $s(t)$ as a parameter of the curve, the following relation holds for a tangent vector:

$$|\dot{\mathbf{P}}(s)| = 1$$

3.4.3 Curvature

The curvature of a space curve cannot be defined in the same manner as a plane curve, and so is defined as follows. First, we move the starting point of tangent vectors to the origin of the coordinate system, while we preserve their direction. Since tangent vectors are unit vectors, the end point of the vectors are on a unit sphere. In lieu of orientation angle $\theta(s)$ used for a plane

curve, the length of the curve traced by the tangent vector is used to define curvature.

$$\theta(s) = \int_{s_0}^{s} \sqrt{\dot{\mathbf{P}}(\sigma) \cdot \dot{\mathbf{P}}(\sigma)} d\sigma$$

Curvature is then defined by taking a derivative of the function $\theta(s)$.

$$\kappa(s) = \frac{d\theta(s)}{ds} = \sqrt{\dot{\mathbf{P}} \cdot \dot{\mathbf{P}}}$$

Curvature for a space curve is non-negative because of the definition.

3.4.4 Torsion

In 2-D, we have a special pair of unit orthogonal vectors: a tangent vector and a normal vector. In 3-D, we have a set of three vectors. The tangent vector $\mathbf{v}_1(s)$ is defined in exactly the same manner:

$$\mathbf{v}_1(s) = \dot{\mathbf{P}}(s)$$

For space curves, the space orthogonal to the tangent vector has two dimensions. In order to single out a normal vector, we assume $\kappa > 0$ or $\dot{\mathbf{P}} \neq \mathbf{0}$ and choose the unit vector $\mathbf{v}_2(s)$ as:

$$\mathbf{v}_2(s) = \frac{\dot{\mathbf{P}}}{|\dot{\mathbf{P}}|}$$

The normal vector thus defined is orthogonal to the tangent vector, because the tangent vector is a unit vector.

The plane determined by the two vectors is called the osculating plane. The third vector $\mathbf{v}_3(s)$ called the binormal vector is defined from the two vectors:

$$\mathbf{v}_3(s) = \mathbf{v}_1(s) \times \mathbf{v}_2(s)$$

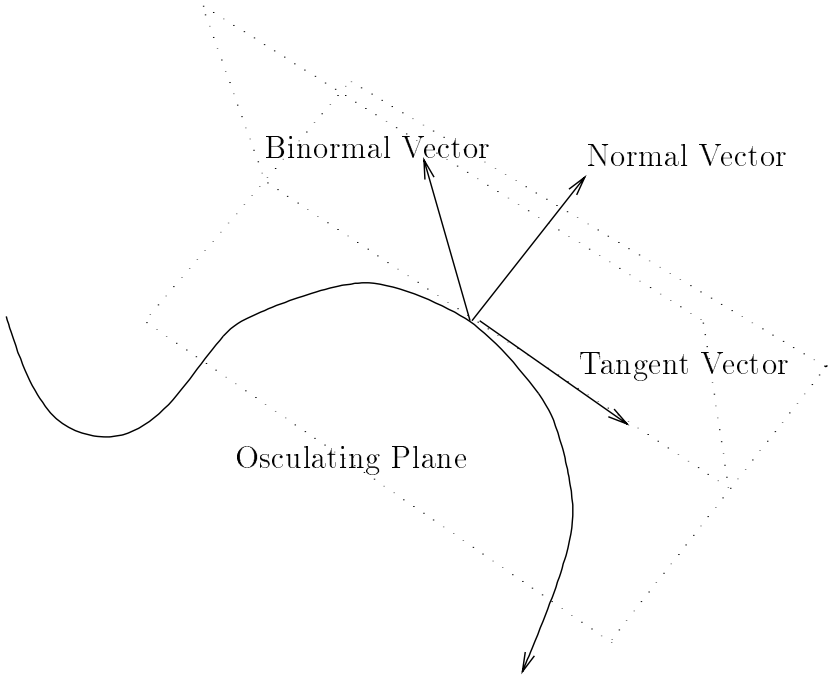


Figure 3.5: Tangent, Normal, and Binormal Vectors

See Fig. 3.5.

For space curves, we have another invariant property of a curve, in addition to curvature. *Torsion* τ is defined by the next equation:

$$\dot{\mathbf{v}}_3 = -\tau \mathbf{v}_2$$

Note that $\dot{\mathbf{v}}_3$ is orthogonal to $\mathbf{v}_1$ and $\mathbf{v}_3$ and thus is parallel to $\mathbf{v}_2$.

3.4.5 Existence of a Space Curve given Curvature and Torsion

The Frenet equations for a space curve are:

$$\begin{pmatrix} \dot{\mathbf{v}}_1(s) \\ \dot{\mathbf{v}}_2(s) \\ \dot{\mathbf{v}}_3(s) \end{pmatrix} = \begin{pmatrix} 0 & +\kappa(s) & 0 \\ -\kappa(s) & 0 & +\tau(s) \\ 0 & -\tau(s) & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1(s) \\ \mathbf{v}_2(s) \\ \mathbf{v}_3(s) \end{pmatrix} \tag{3.9}$$

Using the equations, the following theorem can be shown.

Theorem 2 *For a given continuous function $\kappa(s) > 0$ and $\tau(s)$ defined for $s_0 \leq s \leq s_1$, there is one and only one regular curve (within a rigid motion) such that $\kappa(s)$ is its curvature function, $\tau(s)$ is its torsion function, and s is its arc length.*

Let us make a few comments on the assumption that $\kappa(s) > 0$ in the theorem. The Frenet equations are first order ordinary differential equations. A unique solution exists for the Frenet equations just by assuming $\kappa(s)$ and $\tau(s)$ are continuous, given an initial condition $\mathbf{v}_1(s_0)$, $\mathbf{v}_2(s_0)$, and $\mathbf{v}_3(s_0)$. However, the theorem says something stronger than that, because uniqueness of a solution *as a regular curve* is stated. While the assumption $\kappa(s) > 0$ is necessary for avoiding unnecessary complications in mathematics, we can view the Frenet equations simply as a way to construct any regular curve using two continuous functions $\kappa(s)$ and $\tau(s)$ parameterized by its arc length.

3.5 Continuous Model in 3-D

The continuous model in 3-D is a natural extension of the continuous model in 2-D. For each segment in the model, torsion is considered in 3-D in addition to curvature. The torsion function is also discretized using five points (s_a, τ_a), (s_b, τ_b), (s_c, τ_c), (s_d, τ_d), and (s_e, τ_e). Operators now include those to move τ_a through τ_e. The continuous decomposition technique is extended to 3-D without difficulty.

We use the Frenet equations for space curves (3.9) to obtain a configuration from curvature and torsion. We first apply numerical integration to (3.9) to obtain the orthogonal vectors $\mathbf{v}_1(s)$, $\mathbf{v}_2(s)$, and $\mathbf{v}_3(s)$ from curvature $\kappa(s)$ and torsion $\tau(s)$. Then, the space curve $\mathbf{P}(s)$ is obtained using the tangent

vector $\mathbf{v}_1(s)$. The tangent vector was defined as follows.

$$\mathbf{v}_1(s) = \dot{\mathbf{P}}(s)$$

Hence the space curve is:

$$\mathbf{P}(s) = \mathbf{P}(s_0) + \int_{s_0}^{s} \mathbf{v}_1(\sigma) d\sigma$$

Chapter 4

Solving Open Space Problems

In the previous chapter, the continuous manipulator model was defined. In this chapter, we solve open space problems for a single segment of the continuous model. The open space problem is to find a sequence of curvature and torsion operators for a single segment to reach a goal with its tip in an unobstructed working environment. The open space problem can be seen as the *inverse kinematics* problem (see Section 4.4.1) for a segment of the continuous model. The result of this chapter will be extended in the next chapter, where we define four basic motion schemas to control the continuous model in both open and cluttered space.

Section 4.1 defines four types of open space problems. The section also explains the simulator which we built. Section 4.2 and 4.3 solve the open space problems in 2-D. Related works will be reviewed in Section 4.4. The generality of our approach, along with its 3-D extension, is discussed in Section 4.5.

4.1 Open Space Problems

4.1.1 Four Types of Open Space Problems

Table 4.1 shows the four types of open space problems which appear according to the location of the segment in the continuous manipulator.

OSP1 naturally corresponds to the type of goals to be achieved without decomposition. The rest corresponds to subgoals to be achieved by seg-

goal to achieve	constraint	
	with base rotation	without base rotation
tip position	OSP1	OSP2
tip position and orientation	OSP3	OSP4

Table 4.1: Four Types of Open Space Problems

ments which are generated by decompositions. OSP2 corresponds to the type of subgoals and constraints for the last segment, OSP3 for the first segment, and OSP4 for intermediate segments[1]. Only the first segment is allowed to rotate around its starting point (which is actually the base).

4.1.2 The Swan's Neck Simulator

To test our algorithms and techniques, we built a simulator on a Symbolics machine and name it *Swan's neck simulator* after the actuator which has inspired our research. The original simulator handles a neck of one curvature segment and is equipped with basic hill climbing routines to solve the four types of open space problems. Fig. 4.1 shows its simulation window. Displayed in the simulation window are

- a sequence of configurations of a swan's neck.
- a sequence of curvature functions
- a trace table which shows distances to the goal, selected curvature operators, etc.

More recently, the simulator was extended to handle a series of decomposed segments. Currently, the simulator for 2-D has been completed. The

[1]Segments are numbered from the base to the tip.

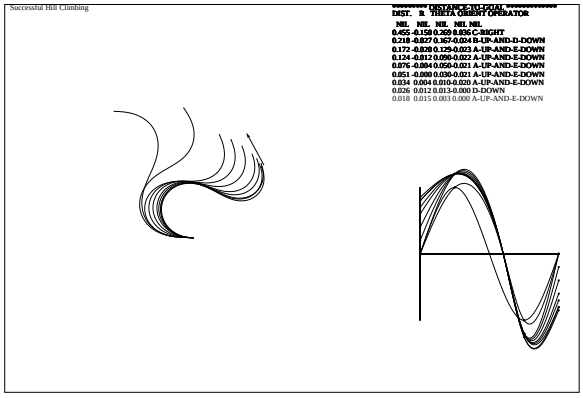


Figure 4.1: Simulation Window

simulator includes a path planning module which finds paths after a user has specified a set of obstacles. Many of the figures in this thesis show output from the simulator.

4.2 Hill Climbing Searches

To solve the open space problems, hill climbing searches are performed in the following sequence.

1. set d^{curr} to the current distance to the goal
2. if $d^{curr} < \epsilon$, exit (the goal has been reached)
3. set $\Delta\tilde{\kappa}$, magnitude of curvature operators, in proportion to d^{curr}
4. choose the curvature operator which makes the next distance d^{next} the shortest, when applied to the current curvature function $\kappa(\tilde{s})$
5. if $d^{next} \geq d^{curr}$, exit (we get caught at a local minimum)
6. update $\kappa(\tilde{s})$ by applying the best curvature operator

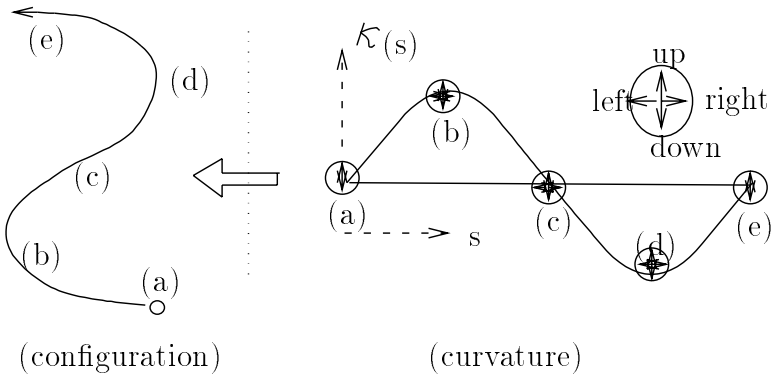


Figure 4.2: Curvature Segment Representation and its Operators. The following *curvature operators* are used to change curvature (and configuration). **a.** Increase/decrease κ_a, κ_b, κ_c, κ_d, or κ_e (move up/down an interpolation point). **b.** Increase/decrease s_b, s_c, or s_d. (move right/left an interpolation point.) **c.** Rotate the base of a curve.

7. update the orientation of the base, if the base can rotate

8. goto 1, and repeat

The curvature operators defined in Section 3.3.1 (see also Fig. 4.2) are used as next state functions. In order to make the magnitude of the curvature operators independent from a curve length L, we use a *normalized curvature function* $\tilde{\kappa}(\tilde{s})$ instead of $\kappa(\tilde{s})$[2].

$$\tilde{\kappa}(\tilde{s}) = \kappa(\tilde{s}) * (L/2\pi)$$

Then the magnitude of the curvature operators is set as below based on experiments.

$$\Delta\tilde{\kappa} = \begin{cases} 0.20 & \text{if } d^{curr} \geq 0.10 \\ 0.10 & \text{if } d^{curr} \geq 0.05 \\ 0.05 & \text{otherwise} \end{cases}$$

[2]Curves with $\tilde{\kappa}(\tilde{s}) = \pm 1$ are circles. $\tilde{s}$ is a normalized curve length parameter and $0 \leq \tilde{s} \leq 1$.

$\Delta\tilde{\kappa}$ is used as an increment/decrement of $\tilde{\kappa}(\tilde{s})$ when we move up/down an interpolation point, and $0.5 * \Delta\tilde{\kappa}$ is used as an increment/decrement of $\tilde{s}$ of $\tilde{\kappa}(\tilde{s})$ when we move right/left an interpolation point.

Distance functions are defined for each type of open space problems as follows.

$$
\begin{aligned}
d_{OSP1} &= |\Delta r|/L \\
d_{OSP2} &= |\Delta r|/L + |\Delta\theta|/\pi \\
d_{OSP3} &= |\Delta r|/L + |\Delta\theta - \Delta\psi|/\pi + |\Delta\phi - \Delta\psi|/\pi \\
d_{OSP4} &= |\Delta r|/L + |\Delta\theta|/\pi + |\Delta\phi|/\pi \\
\Delta r &= r_{goal} - r \\
\Delta\theta &= \theta_{goal} - \theta \\
\Delta\phi &= \phi_{goal} - \phi
\end{aligned}
$$

where

L : curve length

(r, θ) : polar coordinates of the tip of the current configuration

ϕ : orientation of the tip of the current configuration

$(r_{goal}, \theta_{goal})$: polar coordinates of the goal position

ϕ_{goal} : goal orientation

$\Delta\psi$: rotation of the base, which will be explained shortly

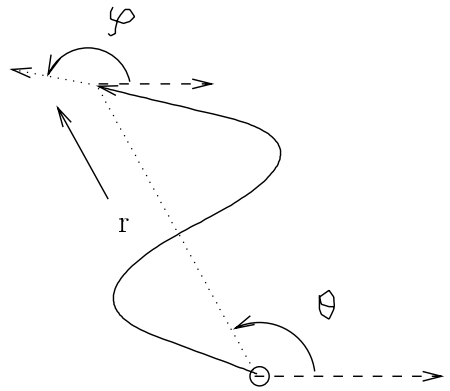


Figure 4.3: Polar Coordinates for Distance Functions

The origin of the polar coordinate system is at the base of the neck (Fig. 4.3). Angles are in radians. When the base is allowed to rotate, its rotation is[3]

$$\Delta\psi = \begin{cases} \Delta\theta & \text{for OSP1} \\ 0.5 \cdot (\Delta\theta + \Delta\phi) & \text{for OSP3} \end{cases}$$

Fig. 4.4 shows an example of a successful hill-climbing search. This and subsequent figures show graphical output from our simulator. Each display shows multiple plots of $(x(s), y(s))$ on the left and $\kappa(s)$ on the right for a finite sequence of times $t_0, \ldots, t_n$. The base of the arrow in the figure represents the goal position, and the orientation of the arrow represents the goal orientation.

4.3 Solving the Local Minima Problem

4.3.1 The Local Minima Problem

OSP1 is easy, because OSP1 has no local minima. All the others have local minima. But the local minima problem is most evident in OSP4 where achieving r_{goal},θ_{goal}, and ϕ_{goal} can compete with one another. Hence, we will

[3]Rotation is chosen to make the distance the shortest.

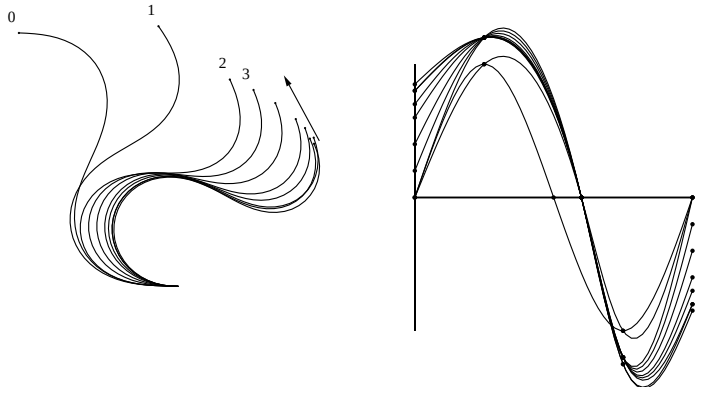


Figure 4.4: Successful Hill-Climbing

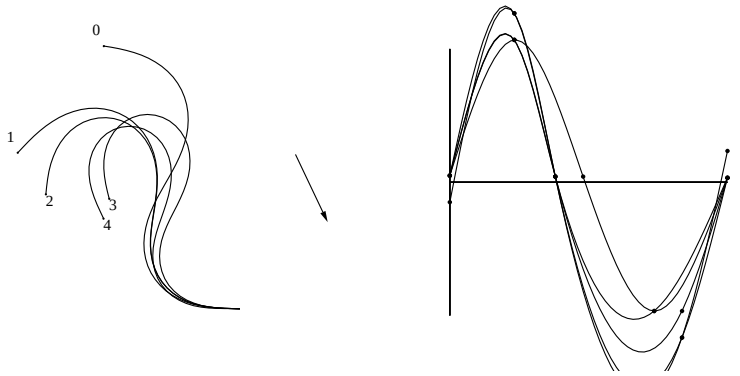


Figure 4.5: Local Minimum in Hill Climbing

talk about OSP4 from now on. Fig. 4.5 is a typical example of getting caught in a local minimum. The naive hill climbing search works in Fig. 4.4 but doesn't in Fig. 4.5.

Besides the local minima problem, the type of the hill climbing search shown in Fig. 4.5 is not good because it usually leads to solutions with configurations which are either self-intersecting or have large curvature. We prefer configurations with small curvature, because they will be easier to approximate

by a jointed arm.

Changing the distance functions, for instance , putting more weight on $\Delta\phi$ in d_{OSP4} to emphasize an orientation, sometimes works, but tends to make another local minimum. We could get rid of some local minima by using *coordinated operators* built on top of the original curvature operators. For example increasing κ_b and decreasing κ_d at the same time in Fig. 4.2 is effective in decreasing the radius r without changing the tip orientation ϕ much, because

- $\phi - \phi_0$[4] is proportional to $\int_0^1 \kappa(\tilde{s})d\tilde{s}$.
- r is inversely proportional to $\int_0^1 |\kappa(\tilde{s})|d\tilde{s}$.

We can thus get a new curvature operator to attain competing goals simultaneously. But in general, coordinated operators are more effective for the fine tuning of configurations rather than for solving the local minima problem. We still have local minima.

4.3.2 Typical Configurations

Let's go back to the examples in Fig. 4.4 and 4.5. Why does the hill climbing search work in one of them and not in the other? In a sense, it is obvious. In the first example, the initial configuration is qualitatively similar to the final configuration. On the other hand, the initial configuration and the goal configuration are qualitatively different in the second example. We can envision the successful final configuration for the second example as an arc-like one with no inflection point.

[4] ϕ_0 is the base orientation.

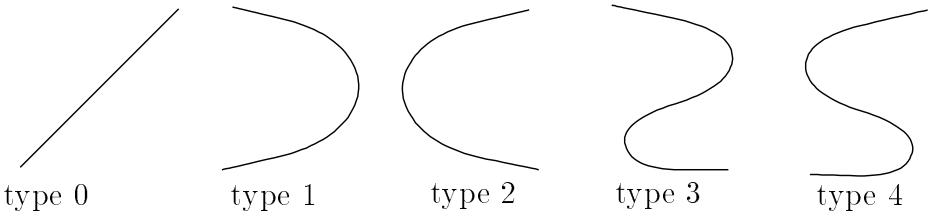


Figure 4.6: Curvature Segment Type

On the basis of the above observation, we tried to coarsely classify the infinitely many configurations by picking up several typical configurations. The typical configurations have been chosen according to the sign(s) of their curvature function. If we limit the number of inflection points within a segment to one, we have only five *curvature segment types* (Fig. 4.6).

In the hill climbing search, given θ, ϕ, and r in Fig. 4.3 are to be achieved. Let us fix the base orientation ϕ_0 to some arbitrary value, say 180°, and consider θ and ϕ ignoring r. The configurations of the five curvature segment types occupies five distinctive places in θ-ϕ plane. See Fig. 4.7. For any point in the θ-ϕ plane, there is a segment type in its neighborhood.

4.3.3 Interpolation to a Good Initial Configuration

We added a capability of finding a good initial configuration for hill-climbing searches. We stored in the simulator five typical configurations which represent the five curvature segment types[5]. The maximum of $\tilde{\kappa}(\tilde{s})$ for these configurations are set to ± 1.

Prior to a hill climbing search, the simulator searches the stored configurations for the best initial configuration. The best configuration is the one

[5] Their curvature along with r,θ, and ϕ are stored.

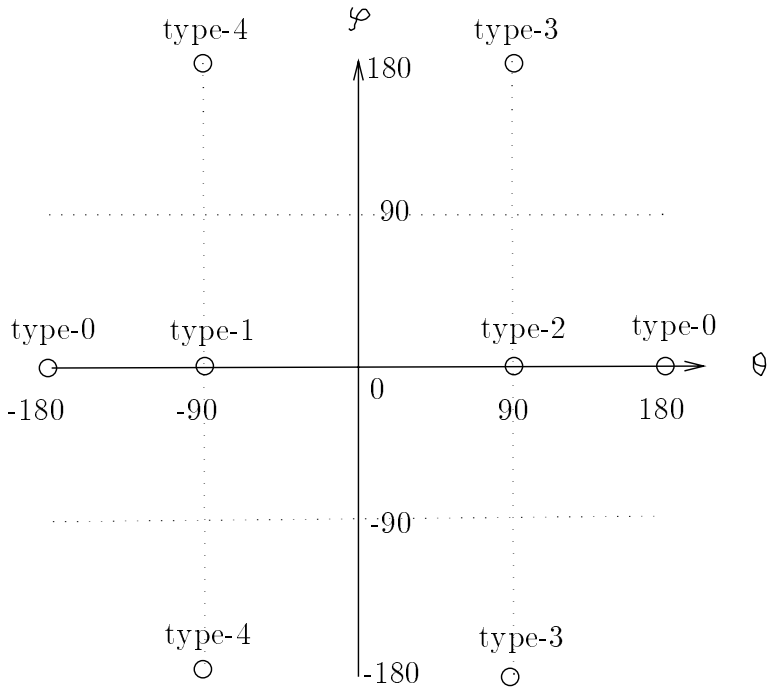


Figure 4.7: Curvature Segment Types in θ-ϕ Plane

which is the closest to the goal in terms of the distance functions which were defined for the hill climbing search, after the stored configurations are scaled and rotated so that they have the same length and base orientation as the segment being simulated (see Fig. 4.8). Here, we need to have a precompiled movement routine to move from the actual initial configuration to the one suitable for hill climbing. Since we know the curvature profile of both configurations, this can be done easily. In order to move from one configuation to another, we simply interpolate the corresponding curvature parameters for the two curvature profiles to obtain the intermediate curvature profiles and the configurations. Fig. 4.9 shows a solution of the same example in Fig. 4.5.

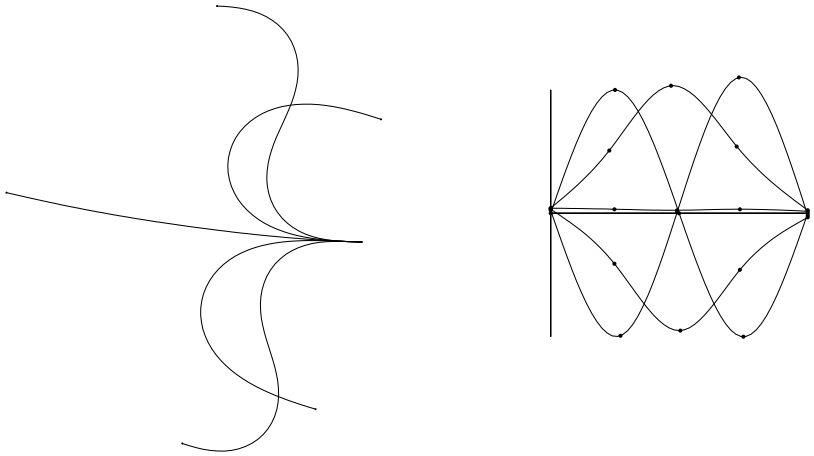


Figure 4.8: Five Candidate Initial Configurations

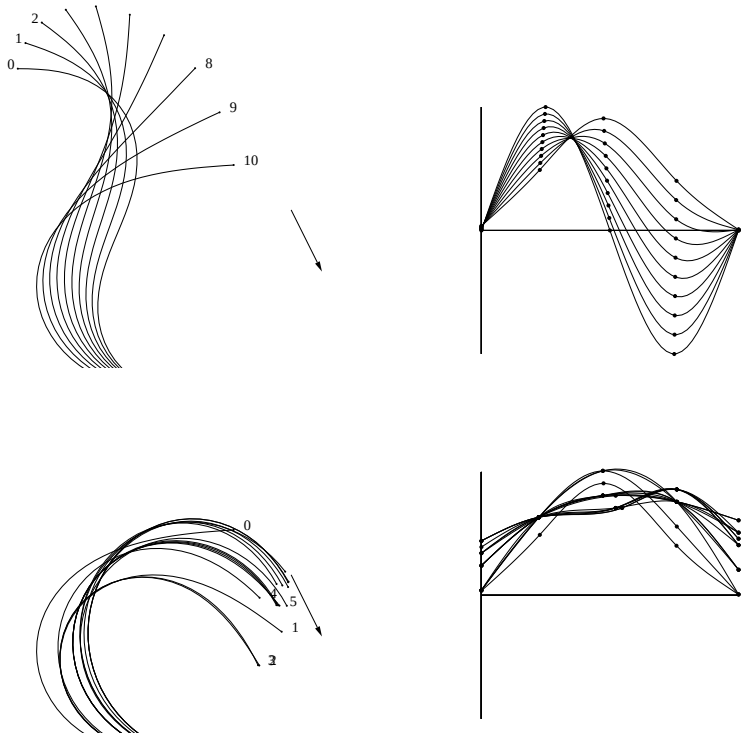


Figure 4.9: Interpolate to Good Initial Configuration and Hill Climbing

4.3.4 Simulation Results

The initial configuration finding capability has turned out to be very effective. We were afraid that we might need lots of configurations as candidates, but it was not necessary. Judging from our experiments, it seems that the five configurations (and a few more at most) are sufficient as far as open space problems are concerned. The simulator sometimes gets caught in local minima, but it happens only when the neck is very close to the goal. We believe this situation can be improved by adding a few coordinated motions or by fine tuning $\Delta\tilde{\kappa}$. We attach at the end of this chapter some of the open space problems solved.

4.4 Related Work on Inverse Kinematics

The open space problem we have solved can be seen as inverse kinematics for a single segment of the continuous model. In this section, we review some of the methods for inverse kinematics developed for redundant manipulators, first for redundant jointed arms and then for continuous arms.

4.4.1 Redundant Jointed Arms

The inverse kinematics problem is defined as follows. Let $\mathbf{q}$ be a vector of joint coordinates, and $\mathbf{x}$ be a vector of end effector coordinates. Then forward kinematics is a mapping $\mathbf{f}$ from $\mathbf{q}$ to $\mathbf{x}$:

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \tag{4.1}$$

The inverse kinematics is to find the inverse of $\mathbf{f}$:

$$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x}) \tag{4.2}$$

For redundant manipulators, inverse kinematics have an infinite number of solutions. An optimization procedure can be used to choose the set of solutions subject to an objective function. Closed form solutions are difficult to obtain for such cases, and most of the research on the problem has been based on a differential representation of (4.1) and (4.2):

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{4.3}$$

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{x}} \tag{4.4}$$

where $\mathbf{J}^{-1}(\mathbf{q})$ is a generalized inverse of the Jacobian matrix J, which can be uniquely determined by taking into account a specified objective function. (4.4) is used iteratively to move the end effector to its goal. The techniques which we discussed in Section 1.4 are examples of this approach, where objective functions are chosen so as to avoid obstacles. [Goldenberg *et al.* 85] generalized these techniques. They reformulated the inverse kinematics problem for redundant manipulators as a constrained nonlinear optimization problem, and used a modified Newton-Raphson method to solve a set of nonlinear equations.

Applying artificial neural network techniques to learning inverse kinematics or inverse differential kinematics has been tried [Josin 88, Ackley 89, Mel 90, Barhen *et al.* 89, Yeung and Gekey 89], and many of these approaches deal with redundant manipulators. [Mel 90] deals with a 3 DOF planar manipulator, [Ackley 89] with a 5 DOF planar manipulator, [Barhen *et al.* 89] with manipulators with many degrees of freedom. One of the advantages of these techniques over numerical optimization techniques would be efficiency, i.e. a rapid convergence to a solution after the network has learned from examples the non-linear transformation $\mathbf{J}^{-1}(\mathbf{q})$.

4.4.2 Continuous Arms

Kinematics of continuous arms has attracted little attention. Representing highly redundant manipulators with a continuous model is itself a quite distinctive approach.

In an attempt to find a good initial configuration for his digital manipulator, [Pieper 68] developed an algorithm to find a curve which is made up of four connected circular arcs in such a way that they have the same radius, and adjacent arcs have the same tangent. The coordinates and the orientation of the tip along with the length of the curve is specified. By expressing the constraints using trigonometric equations, the curve which satisfies the requirements can be found.

[Chirikjian and Burdick 90] presents an approach similar to ours. The authors also represent highly redundant manipulators using a continuous manipulator model. While we use spline interpolation to discretize curvature and torsion, they use modal decompositions. For example, the curvature function can be represented as :

$$\kappa(s) = \sum_{i=1}^{N} a_i \phi_i(s) \tag{4.5}$$

where $\phi_i(s)$'s are called *modes*, and a_i's *participation factors*. In particular, for the primary mode of curvature functions:

$$\kappa(s) = acos(2\pi s) + bsin(2\pi s) \tag{4.6}$$

they have derived a closed form solution to the inverse kinematics problem:

$$(x, y) \mapsto (a, b) \tag{4.7}$$

by using the following formula:

$$\theta(s) \quad = \quad \theta(s_0) + \int_{s_0}^{s} \kappa(\sigma) d\sigma$$

$$\mathbf{P}(s) \quad = \quad \begin{pmatrix} x(s_0) + \int_{s_0}^{s} \cos\theta(\sigma)d\sigma \\ y(s_0) + \int_{s_0}^{s} \sin\theta(\sigma)d\sigma \end{pmatrix}$$

The formula was introduced in Section 3.2 as (3.2). However, limiting curvature functions to the form (4.6) is too restrictive. Moreover, no equation exists for space curves which is equivalent to (3.2). This implies that in general we have to rely on numerical techniques in lieu of closed form solutions.

4.4.3 Research on Curve Design

Somewhat related research, the study of curves which satisfy certain conditions, has been done in computer graphics and applied mathematics.

There is a body of research on designing curves from curvature and torsion, their intrinsic properties [Nutbourne *et al.* 72, Pal and Nutbourne 77, Pal 78a, Pal 78b, Schechter 78a, Schechter 78b]. This research is motivated by the fact that it is often the case that design engineers want to specify the shape of curves in terms of their intrinsic properties, rather than interpolation points or control vertices. For example, piecewise linear curvature functions are used to design plane curves in [Nutbourne *et al.* 72]. [Pal 78a] presents an algorithm to generate a curvature continuous space curve through two points with given tangent directions. Their approach to designing curves through curvature certainly has some appeal to us. But further investigation is necessary in order to extend their technique to the inverse kinematics problem, because curve length cannot generally be specified.

Also interesting is the research on constructing curves which are particularly smooth [Horn 83, Kallay 87, Jou and Han 90]. When the ends of a thin elastic beam are clamped, it will assume a shape which minimizes the

elastic energy:

$$\mathcal{E} = \int_0^L \kappa^2 ds \tag{4.8}$$

Curves which satisfy (4.8) are called *minimum energy splines*. The problem of finding a minimum energy spline given its boundary conditions can be seen as a variation of the open space problems. It has been suggested that humans use this class of curves when they try to complete a partially specified contour map [Horn 83]. [Jou and Han 90] discusses the planar problem, where we have angle constraints at both ends, and a prescribed length for the spline curve. The existence of such splines when the distance between the two end points is within prescribed length is proved, and an iterative algorithm to solve the constraint minimization problem is presented. [Kallay 87] presents an algorithm for the same problem in 3-D.

4.5 Summary of the Chapter

In Section 4.2 and 4.3, we have solved the open space problems in 2-D. In the previous section, we have seen many other methods for inverse kinematics for redundant manipulators. We notice a similarity of these methods to our method. No matter whether the method is intended for jointed arms or for continuous arms, similar procedures are used either in the space of joint variables or in the space of curvature parameters.

We believe that we must resort to an iterative procedure to solve the open space / inverse kinematics problems for highly redundant manipulators. Closed form solutions cannot be obtained except for a very limited class of problems. Possible iterative techniques include a modified Newton-Raphson method or hill climbing search which we use. Learning using neural networks can also classified in this category of techniques.

One advantage of the modified Newton-Raphson method is that an optimality criterion can be incorporated in determining a generalized inverse of the Jacobian matrix through optimization in $n-6$ dimensional space for an n DOF manipulator. However, the method cannot be applied to continuous arms because its Jacobian matrix cannot explicitly be obtained. On the other hand, hill climbing search can be applied to the continuous manipulator model, and our hill climbing search is still in an 8-D space (the space of curvature operators) for a single segment. Furthermore, path planning will become much easier by using the continuous model, as we will see in the following chapters. The local minima problem is inherent to both of the techniques, and starting from an initial value close to the solution is necessary in order not to be caught at local minima. We have also addressed the problem of finding a good initial value for our hill climbing search.

While the techniques we have developed are for solving 2-D problems, 3-D problems can be solved by extending the techniques in a straightforward fashion. It will be easy to extend our hill climbing search so that torsion parameters are also be changed to reach a goal in 3-D space. The local minima problem for the extended hill-climbing search will be dealt with in the same way, by storing typical configurations as candidate initial states for the search. Experiments in 3-D are necessary to find a set of typical configurations adequate to avoid local minima.

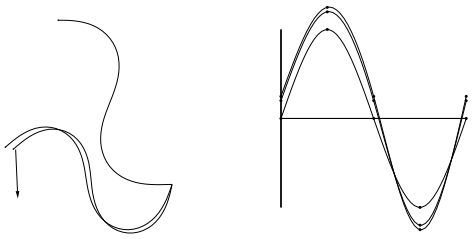


Figure 4.10: Example 1 as OSP1

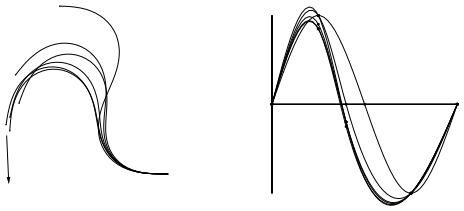


Figure 4.11: Example 1 as OSP2

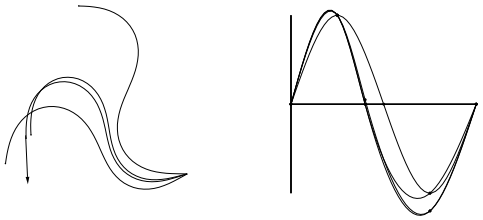


Figure 4.12: Example 1 as OSP3

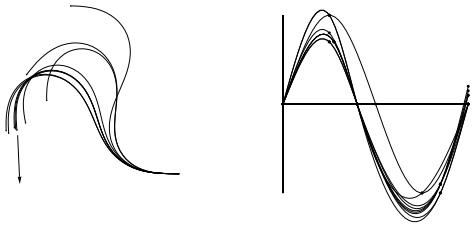


Figure 4.13: Example 1 as OSP4

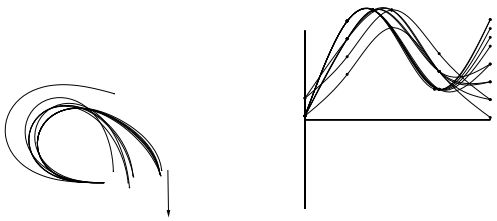


Figure 4.14: Example 2 as OSP4

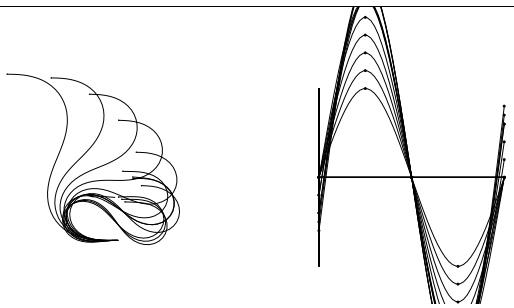


Figure 4.15: Example 3 as OSP4

Chapter 5

Basic Motion Schemas for Path Planning

In the open space problems discussed in the previous chapter, we are concerned only about the *tip* position/orientation to reach a goal, hence we need only one segment. To solve a more complex problem, the manipulator can be segmented at points along the length of the manipulator, and individual *motion schemas* are combined to achieve multiple subgoals along its length.

In this chapter, we define basic motion schemas that control an individual segment of the continuous manipulator. We express a motion plan for the continuous manipulator using decompositions and motion schemas. By executing the plan, the manipulator trajectory will be obtained. The path planning problem for the continuous manipulator model is analyzed, and our approach to the problem is introduced.

5.1 The Basic Motion Schemas

We define four basic motion schemas that control an individual segment:

Hill-climb: Hill climb to achieve tip position and/or orientation or end curvature/torsion.

Interpolate: Move by interpolation between two specified curvature/torsion profiles.

Feed/Retract: Increase (decrease) the length allocated to a segment by moving the tip along a trajectory to reach a given position/orientation.

Fold/Unfold: Increase (decrease) the length allocated to a segment, while maintaining tip position and orientation.

The first two motion schemas are defined to solve the open space problems. The other are added to control a segment in cluttered space. Using the decomposition technique, the motion schemas can be combined to control the whole manipulator. Details follow on each motion schema we have implemented for the 2-D simulation.

5.1.1 Motion Schemas for Open Space

A *Hill-climb* schema executes a set of hill-climbing routines to achieve the four types of open space problems. Distance functions for hill climbing search are defined based on the goal, the tip and the rotation of the base. The curvature operators are used as next-state functions with magnitude scaled in proportion to the current distance to the goal. To have continuous curvature in decompositions, another hill climbing routine has been added to achieve end curvature while maintaining its tip position and orientation.

An *Interpolate* schema moves by interpolation between two specified curvature profiles. Each of the curvature parameters, (s_a, κ_a), (s_b, κ_b), (s_c, κ_c), (s_d, κ_d), and (s_e, κ_e), is linearly interpolated.

$$\begin{aligned} s_*^{interpolated} &= (1-p)s_*^{from} + ps_*^{to} \\ \kappa_*^{interpolated} &= (1-p)\kappa_*^{from} + p\kappa_*^{to} \end{aligned}$$

where p moves from 0.0 to 1.0. The naive hill climbing search does not always work, because it may get caught at a local minimum. Hill climbing search works

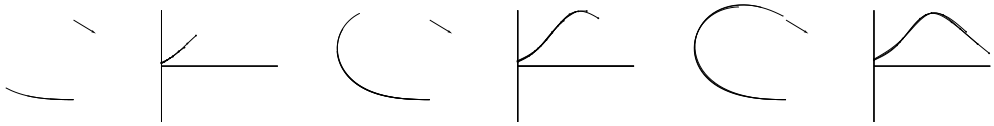


Figure 5.1: Feed/Retract Motion Schemas

when it starts from a configuration close to the final configuration. Hence, we select and move to a good initial configuration before hill-climbing search. *Interpolate* motion schema moves from the actual initial configuration to the selected configuration.

5.1.2 Motion Schemas for Cluttered Space

Feed/Retract motion schemas increase (decrease) the length allocated to a segment by moving the tip along a trajectory to reach a given position/orientation (Fig. 5.1). This is a motion schema to represent the follow-the-leader type, snake-like motion considered in [Clement and Iñigo 90] which was introduced in Section 1.2. The trajectory is selected from those which correspond to the five typical configurations and cubic spiral curves (see Chapter 6). Since the configuration (shape) of a segment is controlled by five points to represent curvature, the curvature profile of the specified trajectory is traced (by the five points) rather than the trajectory itself. The trajectory determines how precisely we can feed or retract a segment along the trajectory.

Fold/Unfold motion schemas increase (decrease) the length allocated to a segment, while maintaining tip position and orientation (Fig. 5.2). This motion schema is implemented as another kind of hill climbing search. The length for the segment is increased (decreased) incrementally step by step.

For each step, we perform a hill climbing search to maintain the tip position/orientation.

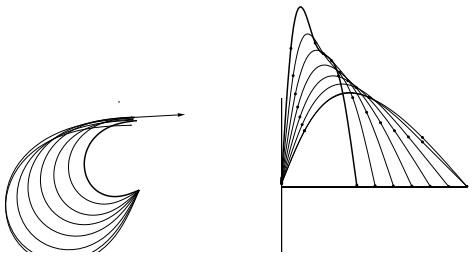


Figure 5.2: Fold/Unfold Motion Schemas

5.2 Using Motion Schemas with Decomposition

A motion plan is expressed using the basic motion schemas. Figure 5.3 shows the structure of a plan. The solution to the local minima problem in open space (figure 4.9) uses two motion schemas, *Interpolate* and *Hill-climb*, after selecting an appropriate curvature segment type. The manipulator can be decomposed to achieve multiple subgoals along the length of the continuous manipulator. Each subgoal is then achieved by executing motion schemas for the corresponding segment. Four decompositions have been implemented: *add/delete* a segment, *divide* a segment at a specified point, and *merge* two

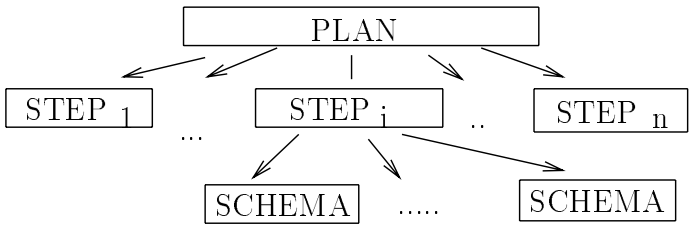


Figure 5.3: Motion Plan Representation

neighboring segments.

The motion plan is then executed to obtain a trajectory. Use of the motion schemas with decompositions is demonstrated in figure 5.4, where we fold, rotate, and extend the manipulator.

To achieve a set of subgoals along the length of the manipulator, Frames 7-10 in Fig. 5.4 are repeated:
For each of the subgoals, the following steps are executed.

1. Add a segment at the tip of the continuous manipulator. The initial length of the added segment is zero.

2. Feed the new segment incrementally while unfolding the segment of the manipulator near the base by the same increment of length. Please refer to Fig. 5.4 on the details of the motion schemas used to unfold the manipulator.

   The next two steps are necessary, because the curvature profile of the path segment is approximated using the five points.

3. Hill climb for the tip segment to achieve the subgoal position precisely.

4. Hill climb to set end curvature to zero while maintaining the subgoal position.

5.3 Path Planning Problem for the Continuous Manipulator

We have solved the open space problem. We have also demonstrated the use of decompositions and motion schemas. Now let us combine these results and attack the whole problem: *given a highly redundant manipulator*

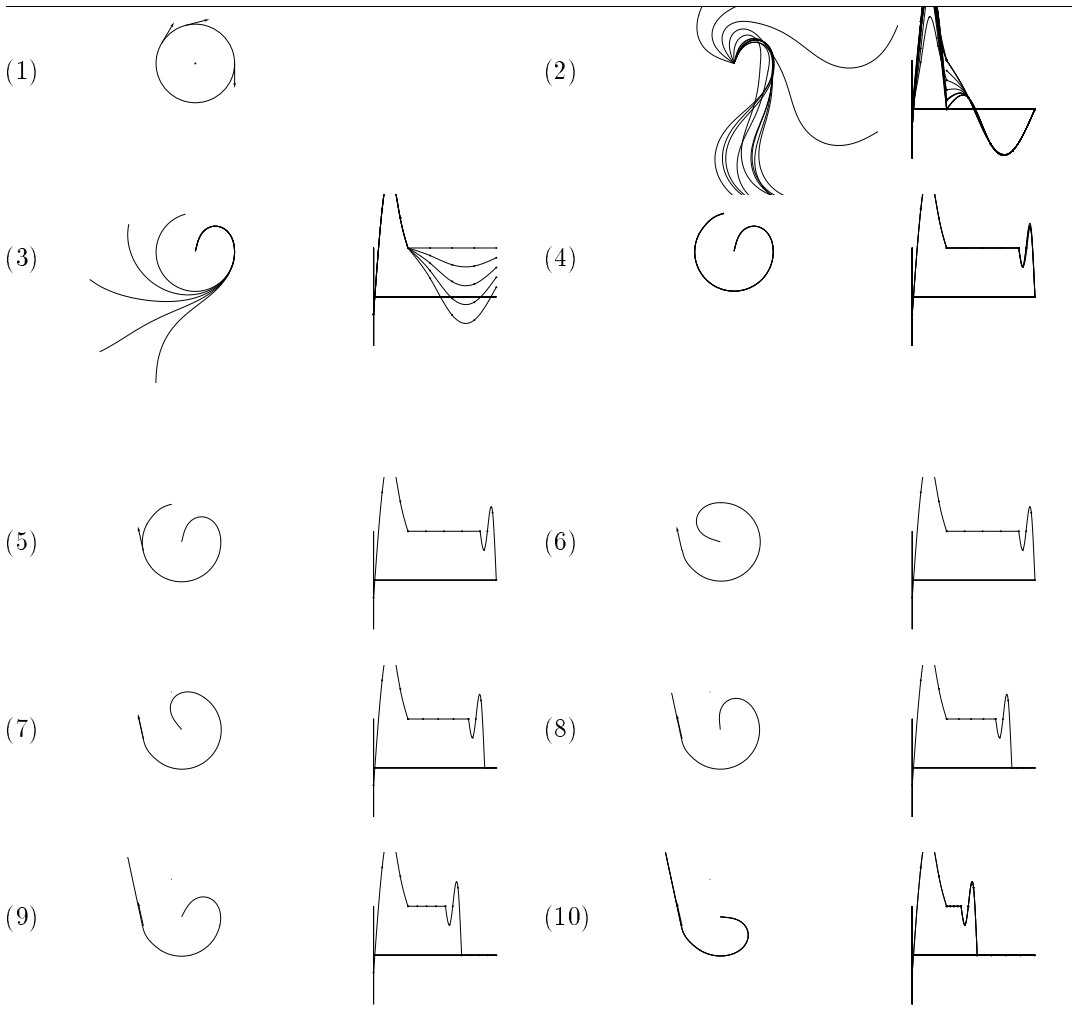


Figure 5.4: Fold, Rotate, and Extend a Manipulator: **Frames 1-4:** Given a radius r, we fold the manipulator into a circular arc configuration. **(1)** The circle with vectors shows the subgoals for the three segments. **(2)** We divide the manipulator into two segments and achieve the first subgoal using *Interpolate* and *Hill-climb*. **(3)** Then we use *Interpolate* with curvature $1/r$ for the second segment. **(4)** Finally, we divide the second segment and use *Hill-climb* to set the end curvature to 0. **Frames 5-10:** We then rotate and extend the manipulator to pass through two new subgoals. **(5)** The two new subgoals are shown. **(6)** Rotating the manipulator can be done by *Hill-climb* for the first segment with its subgoal rotated. **Frames 7-10:** We add the fourth segment to extend the manipulator. The target trajectory for the segment (a straight line) is selected, and then *Feed* (fourth segment), *Unfold* (second segment) and *Hill-climb* (first segment) are combined to extend the manipulator incrementally. (See also Fig. 7.2.)

and its environment, how can we find a collision free trajectory automatically and efficiently?. Figure 5.5 is an overview of a motion planning system and the area enclosed by dotted lines are what we have explained so far. The dynamic simulation is explained in [Park 90].

5.3.1 Where Do We Start?

The following proposition shows where we start.

Proposition 1 *Suppose we have a continuous curvature manipulator with variable length. Then, given polygonal obstacles, we can find a collision free path from a current configuration to the goal configuration. The time complexity is polynomial in the number of obstacle corners.*

Proof (sketch). First, we find a configuration to reach the goal from the base. For a *point robot*, there is an efficient algorithm called *visibility graph method* to find the shortest path in 2-D space which is cluttered with polygonal obstacles. A visibility graph is constructed by making both obstacle corners and start/end points nodes of the graph and by connecting those nodes which are visible from each other. The shortest path can be found by searching through the visibility graph. See [Sharir and Schorr 84] for a sketch of the method. The problem can be solved in $O(n^2 \log n)$ time where n is the number of obstacle corners. The algorithm has been improved to $O(n^2)$ time [Asano 85, Welzl 85].

Then it is easy to transform the polygonal path to a one which is also collision free and has smooth corners (has curvature continuity). We can apply the method in [Kanayama and Hartman 89] which uses cubic spiral curves for

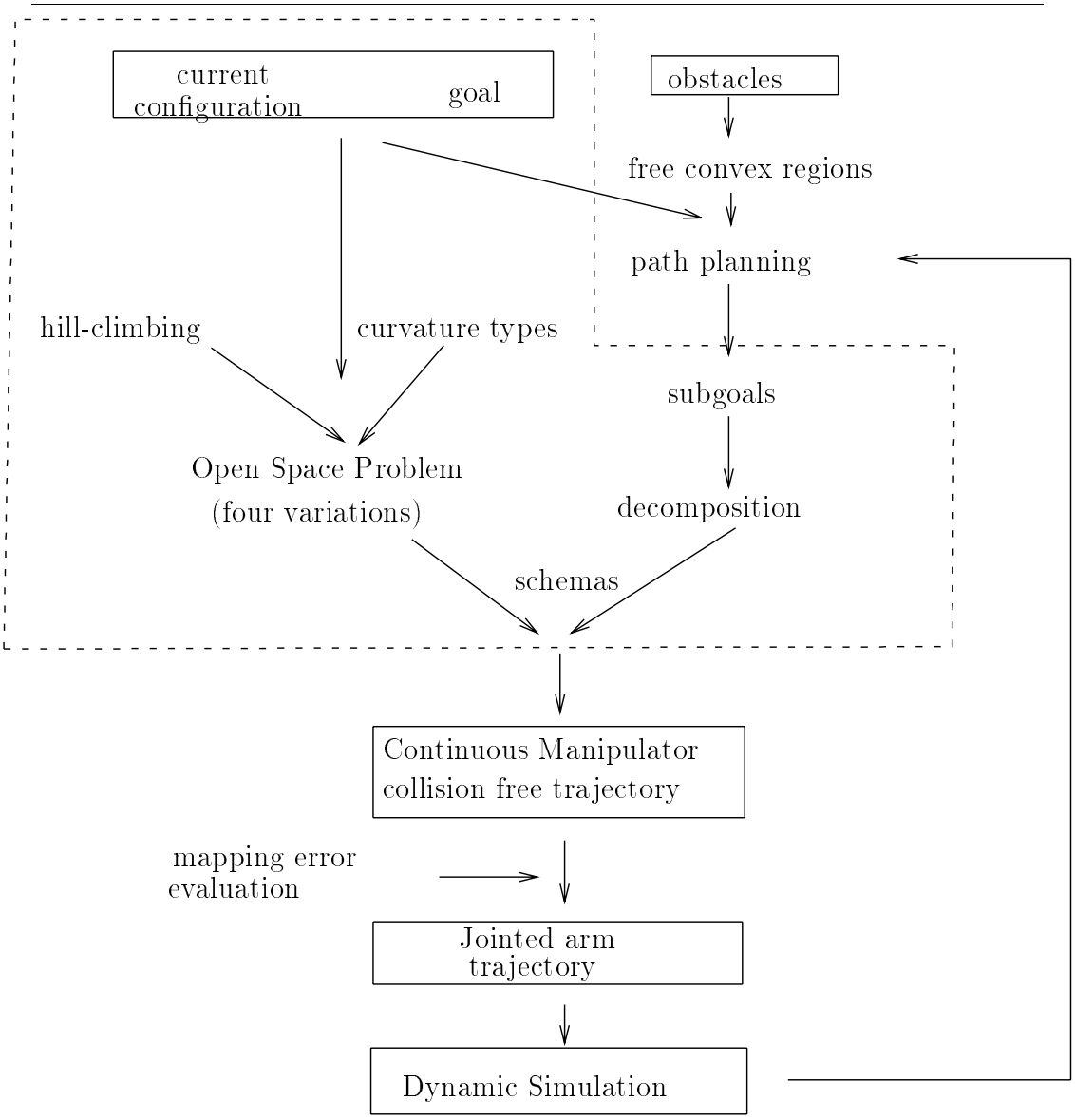


Figure 5.5: Overview of the Motion Planning System: the area enclosed with dotted lines has been explained so far.

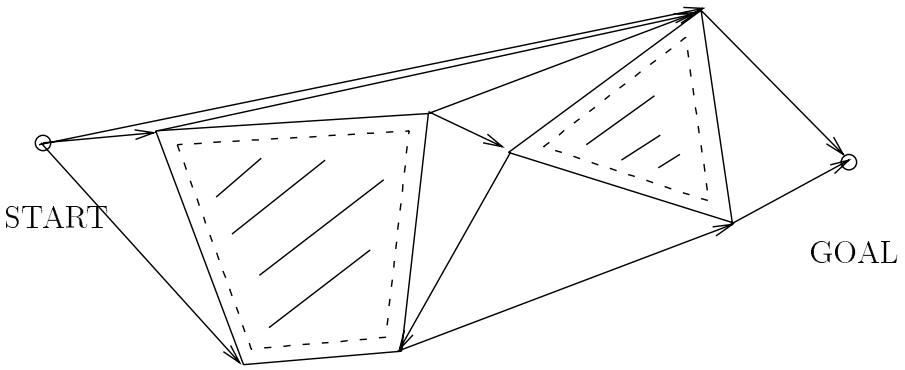


Figure 5.6: Visibility Graph Method To Find a Polygonal Path. Obstacles boundaries are drawn in dotted lines and are grown to make room for turns.

turns to maintain curvature continuity[1]. By growing the obstacles a little in advance, we can find enough space to make small smooth turns.

We are going to retract the manipulator to a point at the base, and then extend the manipulator to the goal along the continuous curvature path found. A little bit care must be taken, because our continuous curvature manipulator model is controlled by the curvature parameters, and hence cannot precisely follow the curvature profile of the path found. To obtain the curvature profile for the continuous manipulator, we use motion schemas.

We assign a unit cost to each motion schema execution. The unit cost assignment is justified because the task here is considered to be a control task, rather than a planning task as follows.

- The unit cost does not include the cost for planning. A continuous curvature path for a point robot has already been found, consisting of cubic spiral segments.

[1] Cubic spiral curves will be explained in the next chapter.

- The objective for each motion schema execution is to trace each of the above cubic spiral segments with a single segment of the continuous model.

- The curvature profile of a cubic spiral curve is a quadratic function of length s, simple enough to be traced (interpolated) by a single segment of the continuous model with its five curvature parameters adjusted. In fact, iterations for adjusting curvature parameters converge rapidly.

We first retract the whole length of the manipulator and make it a point at the base. If the current configuration has m curvature segments, this takes $O(m)$ steps using decompositions and schemas.

Following the above smooth path is just a reverse process of retracting the whole manipulator. Since the shortest path between two given points must be a polygonal line whose vertices are corners of obstacles, the total number of the straight line segments of the path is $O(n)$. We need one curvature segment for each turn. Then the total number of curvature segments we need is $O(n)$, including those for turns. This also shows that we can assume $m = O(n)$. □

5.3.2 Remaining Problems

The previous proposition is important because it shows that the algorithm is dependent on the complexity of the environment rather than the degree of freedom of the manipulator, thanks to the continuous manipulator model. The property comes from the ability of the continuous manipulator to follow its tip trajectory.

But it is not a complete solution, because we are neglecting the following considerations.

[**Manipulator Length**] Manipulators have fixed length.

[**Trajectory Finding**] This is a problem of changing from a current configuration to a final configuration without colliding obstacles. Retracting the whole and reaching from the base is not possible for fixed length manipulators.

[**Maximum Curvature**] In the proposition, we did not put any limitation on curvature values. Kokkinis and Wilson studied the hardware architectures of continuous robotic arms in [Kokkinis and Wilson 88], and found that the maximum curvature is one of the most important performance measures. We need to find a smooth path whose curvature does not exceed a given maximum value.

[**Evaluating Errors of Mapping to a Jointed Arm**] Intuitively, configurations (curves) with large curvature are more difficult to approximate using a jointed arm. We need to evaluate the upper bound of the mapping errors, given a jointed arm and the maximum curvature. Then, we can "grow obstacles" by an appropriate amount in advance so that a collision free configuration for the continuous manipulator within the maximum curvature is guaranteed to be mapped to a collision free configuration for a jointed arm (see Fig. 5.7).

5.3.3 Our Approach

The first two problems are difficult. As to the manipulator length control, think of this. There are efficient algorithms to find the shortest path or nearly the shortest path in 2D. But finding a path with the desired length seems to be much harder. For example, a somewhat similar discrete problem:

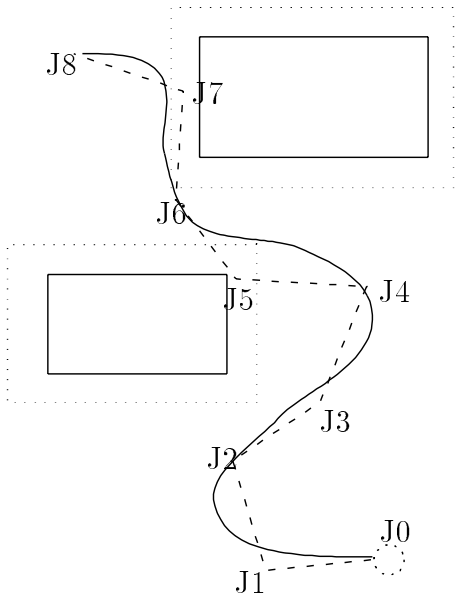


Figure 5.7: Grow Obstacles

the longest path problem for a weighted graph is proved to be NP complete [Garey and Johnson 79].

In order to turn around the problems, we assume some open space around the base to fold/unfold the manipulator. With the assumption, the length is controled by folding/unfolding the manipulator, and the trajectory can be found simply by retracting/extending the manipulator. We present an algorithm to find a path with maximum curvature constraint in Chapter 6,7. In Chapter 8, we show how to set the maximum curvature constraint, given a jointed arm.

Chapter 6

Planning a Smooth Path for Autonomous Vehicles using Primary Convex Regions

This chapter is about path planning for autonomous vehicles. It is included, because the algorithm to find smooth paths for autonomous vehicles can be directly applied to path planning for the continuous manipulator model.

This is a self contained chapter. First, the problem of finding smooth paths is introduced. Then, we present an algorithm to find smooth paths. Our algorithm is based on decomposing free space into primary convex regions, and is a natural extention of previous algorithms which find straight line paths.

6.1 Introduction

Smoothness of path is critical for autonomous vehicle navigation. Trying to follow unsmooth paths leads to "stop,turn, and move" or "backing up maneuver" strategies. There are algorithms to find a path which consists of straight line and tangent circular arc segments [Wilfong 88, Wilfong 89, Jacobs and Canny 89, Basu and Aloimonos 90]. However, curvature discontinuity exists at every tangent point in these paths, because circular arcs have constant curvature equal to the inverse of their radius and straight lines have zero curvature. This type of path is criticized for not being smooth enough for autonomous vehicles to follow easily [Kanayama and Hartman 89, Nelson 89]. For example, spline curves are recommended over circular arcs for highway transition curves [Barnett 38].

[Kanayama and Hartman 89] proposes using cubic spiral curves to make a smooth (i.e. continuous curvature) move from one position and orientation to another. Cubic spirals can be constructed to have zero curvature at tangent points. But they did not address the path planning problem.

We present a path planning algorithm to find a smooth path which consists of straight lines and cubic spirals. The maximum curvature of cubic spirals is restricted to allow for the vehicle's constraints. Our algorithm is based on decomposing free space into primary convex regions [Rueb and Wong 87, Singh and Wagh 87]. We naturally extend previous algorithms which find straight line paths. Overlapping regions of the primary convex regions are used to make smooth turns from one region to another. Large overlapping regions can be further divided in order not to miss smooth turns. Because of the convex nature of free regions, we can adjust the places of turns easily while keeping a path within free space. We find the shortest smooth path using standard graph search techniques for the connectivity graph which is built on top of the representation.

The rest of the chapter is organized as follows. Section 6.2 concerns free space decomposition into convex regions. Conditions for making a smooth turn between regions are made explicit in section 6.3. Section 6.4 discusses building a connectivity graph of regions and the search for a smooth path.

6.2 Free Space Decomposition

We assume obstacles are given as a set of polygons. We first decompose free space into convex regions using existing methods in literature. It is claimed that path planning algorithms using free space decomposition are better suited for navigation purposes than configuration space algorithms, because

free convex regions can be used to locate and guide the robot locally within the regions by measuring its distance to the region edges[1]. This eliminates the requirement for an accurate global coordinate system to track the position of the robot in navigation. [Rueb and Wong 87].

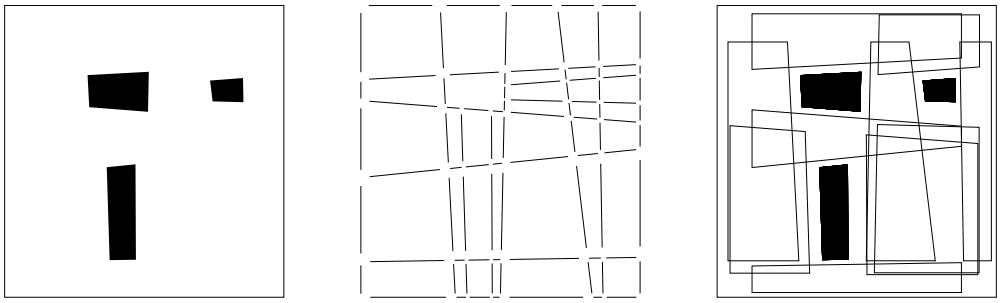
6.2.1 Free Space Decomposition Methods

We need to select a primitive for free space decomposition best suited to our task: finding smooth paths.

[Brooks 83a] is one of the earliest papers on free space decomposition for path planning. In his algorithm, free space is decomposed into generalized cones which are considered to be free ways. But the robot is required to pass along the spines of generalized cones, which leaves less room for our path optimization algorithm to make smooth turns. Moreover, since generalized cones are not convex regions in a strict sense, it follows that if we change the position of a corner within an intersection of two cones, the located line segment is *not* guaranteed to be within the cone.

[Kuan *et al.* 85] decomposes free space into two sets of disjoint polygons: passage regions and channel regions. Passage regions correspond to open space (rooms, squares,..), and channel regions to paths connecting open space. Passage regions may be good candidates for smooth turns. However, a large open space does not necessarily become a passage region, because passage regions and channel regions are determined only from the topology of obstacle layout without considering the size of areas. Moreover, since this algorithm de-

[1] This does not apply to all the free space decomposition methods. Only those decomposition methods to convex regions whose edges correspond to actual walls have this property.



(1) (2) (3)

Figure 6.1: Wall Segments and Primary Convex Regions. **(1)** Obstacles and boundaries. **(2)** Wall segments are shown as line segments. **(3)** Each primary convex region is shrinked for visibility.

composes free space into *disjoint* convex regions, it does not recognize straight line path segments if they exist, so that the path obtained will have more line segments (and more turns) than paths obtained by other methods.

6.2.2 Primary Convex Regions

[Singh and Wagh 87, Rueb and Wong 87] use *primary convex regions* as a primitive to represent free space, in order to find polygonal paths. A primary convex region (PCR) is an unobstructed convex region with each boundary edge covering some portion of an obstacle wall (See figure 6.1).

Since each edge of a primary convex region covers some portion of an obstacle wall, the region seems to be a natural description of the open space bounded by obstacles. In addition, PCR can be seen as an extension of the passage regions described in [Kuan *et al.* 85]. In fact, for any passage region, there exists some primary convex region which contains the passage region. PCRs are suitable representation for our task for the following reasons:

- PCRs are maximal (in area) convex regions surrounded by obstacle edges.

- The convexity of PCRs makes the path optimization easier. If we change the position of a corner within the intersection of two PCRs, the moved line segments are guaranteed to be still within the two PCRs.
- The intersections of PCRs are also convex regions, which is a desirable property.
- Since free space is decomposed into intersecting PCRs, it is easier to find a straight line path segment, and hence obtain paths with fewer turns.

Use of primary convex regions to represent free space has renewed much attention recently. An algorithm to detect primary convex regions from an input image was presented in [Tokuta and Hughes 90]. [Habib and Yuta 89] experimented with Singh and Wagh's method using an actual mobile robot and reported good results.

6.2.3 Hypergraph Method for Finding PCRs

We have implemented the algorithm in [Rueb and Wong 87] to find PCRs given obstacles. PCRs are found by a directed search for a set of fundamental circuits in an abstract graphical representation of the environment geometry. The nodes in the graph are *wall segments* obtained by extending obstacle walls (see Fig. 6.1).

Let n be the total number of obstacle walls. The total number of wall segments is then $O(n^2)$. Rueb and Wong's algorithm to find fundamental circuits in a graph runs in time proportional to the square of the number of nodes in the graph. Hence the complexity of their algorithm is $O(n^4)$. We can also obtain an upper bound for the total number of primary convex regions: $O(n^4)$.

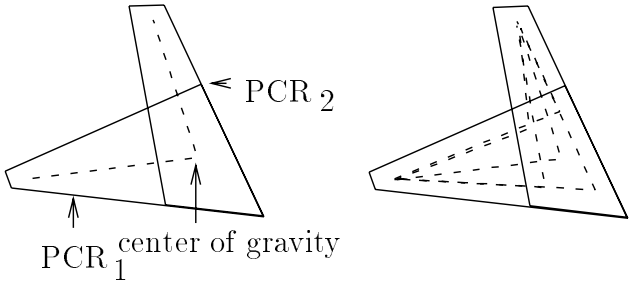


Figure 6.2: Candidate Turning Corners

Although the worst case complexity of the algorithm is not favorable, the algorithm is efficient in practice. Rueb and Wong reported an $O(n)$ performance result (compared to $O(n^4)$ for the worst case analysis). This is because in reasonable environments the total number of wall segments is linear in the number of wall edges; and because their algorithm to find fundamental circuits usually runs in linear time (in the number of nodes) by employing some heuristics.

6.3 Making a Smooth Turn between PCRs

6.3.1 Candidate Turning Corners

Finding a polygonal path is easy, once PCRs are obtained. To move from one PCR to another PCR, a turn can be made anywhere within the intersection of two PCRs.

However, in order to make a smooth turn from one PCR to another PCR while satisfying the maximum curvature constraint, we need to locate *turning corners* appropriately. This is necessary also for finding a shorter path to reach a goal. For a small overlapping region, we use its center of gravity as a turning corner. A large overlapping region can be further divided around its center of gravity in order not miss smooth turns. See figure 6.2. Due to the

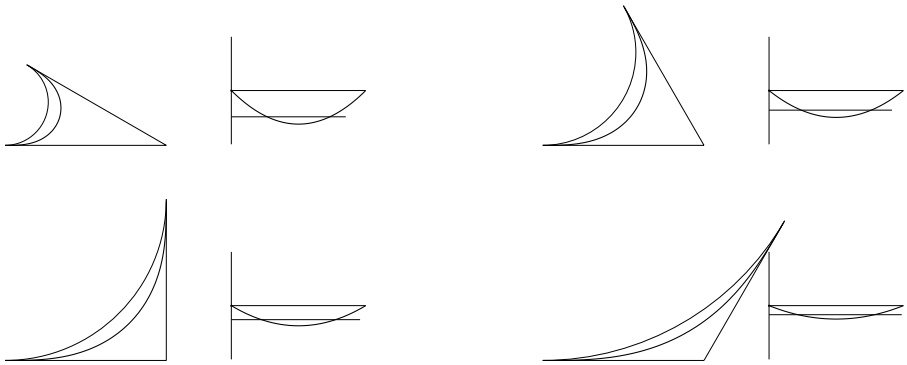


Figure 6.3: Cubic Spirals and Circles: Curves outside are tangent circles, while those inside are cubic spirals (in left figures). While circles have constant curvature, cubic spirals have quadratic curvature functions (in right figures).

convexity of PCRs, the line segments are guaranteed to be within the PCRs.

6.3.2 Cubic Spirals

We use cubic spirals to provide a continuous curvature path, since they can be constructed to have zero curvature at tangent points. A cubic spiral is a curve whose orientation (integration of the curvature) is described by a cubic function of path distance s. See figure 6.3 to see the difference between cubic spirals and circles.

Other spline curves such as Bezier curves or B-spline curves (see [Farin 90]) share some of the favorable properties of cubic spiral curves. Bezier curves and B-spline curves have the convex hull property which makes it possible to fit generated curves within free space by choosing control vertices appropriately. They can be constructed to have zero curvature at end points, too. However, the following properties of cubic spiral curves make them especially suitable for the mobile robot navigation problem.

- Cubic spiral curves are particularly smooth in the sense that they minimize the following cost [Kanayama and Hartman 89].

$$\int_0^l (\dot{\kappa}(s))^2 ds$$

The cost represents the variation of the instantaneous centripetal acceleration (or jerk), because the acceleration is proportional to curvature.

- Maximum curvature is more easily controlled for cubic spiral curves than other spline curves because of their resemblance to circular arcs (see Fig. 6.3). It has been proved that the shortest path in 2-D with a maximum curvature constraint (without the curvature continuity requirement) consists of circular arcs and straight lines [Dubins 57].

6.3.3 Making Smooth Turns using Cubic Spiral Curves

[Kanayama and Hartman 89] presents a method to make a smooth move from one position and orientation to another, using cubic spiral curves .

Proposition 2 (Kanayama and Hartman) *If the size d and the deflection α of a cubic spiral is given (figure 6.4), its length l, curvature κ are*

$$l = \frac{d}{D(\alpha)} \tag{6.1}$$

$$\kappa(s) = \frac{6\alpha D(\alpha)^3}{d^3}((\frac{l}{2})^2 - s^2) \tag{6.2}$$

where

$$s \in [-\frac{l}{2}, +\frac{l}{2}]$$

$$D(\alpha) = 2\int_0^{1/2} \cos(\alpha(3/2 - 2s^2)s)ds$$

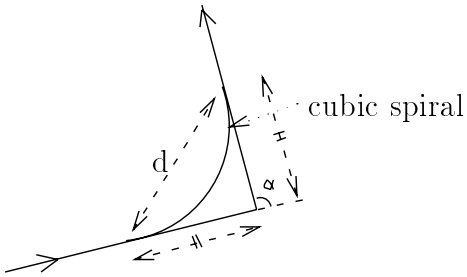


Figure 6.4: Making a Smooth Turn using a Cubic Spiral

This result is directly applicable to making a smooth turn. For each candidate corner, we check whether we can make a turn as follows.

1. Find d_{min}, the minimum d consistent with the maximum curvature constraint.

2. Find d_{max}^{free}, the maximum d for which the curve lies entirely within free space.

3. Find d_{max}^{fit}, the maximum d for a cubic spiral to fit along both tangent line segments.

4. Check $d_{min} \leq min(d_{max}^{free}, d_{max}^{fit})$. This guarantees that we can make a collision free turn within the maximum curvature.

In order to find d_{min}, note that $\kappa(s)$ in (6.2) has its maximum at the midpoint:

$$\kappa_{max} = \kappa(0) = \frac{1.5\alpha D(\alpha)}{d} \tag{6.3}$$

Therefore,

$$d \geq d_{min} = \frac{1.5\alpha D(\alpha)}{\kappa_{max}} \tag{6.4}$$

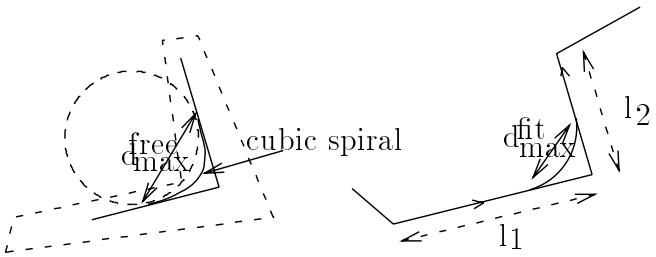


Figure 6.5: d_{max}^{free} and d_{max}^{fit}

To find d_{max}^{free} is not easy, because cubic spirals are expressed via curvature. However, a cubic spiral is always contained in the area outlined by its tangent lines and the circular arc which is tangent at the same points. Such circles are shown in figure 6.3 with cubic spirals. To find a tangent arc which is both collision free and has the maximal radius r_{max}, we apply the condition that the arc passes through one of the corners of the overlapping region (figure 6.5), and obtain:

$$d \leq d_{max}^{free} = 2r_{max}sin(\alpha/2) \tag{6.5}$$

It is possible to find whether we can fit smooth turns by using d_{min} and d_{max}^{free} obtained, *given a whole candidate polygonal path.* However, this leads to an exhaustive search, because each turn affects the preceding and following turns. To avoid an exhaustive search, we use a local fit method. When making a turn, we confine its starting/ending point to within a distance of $l_{min} = min(l_1/2, l_2/2)$ from the turning corner, where l_1 (l_2) is the length of a incoming (outgoing) line segment (figure 6.5). To make a turn within l_{min}, d must satisfy the following:

$$d \leq d_{max}^{fit} = 2l_{min}cos(\alpha/2) \tag{6.6}$$

6.4 Graph Search for a Smooth Path

The previous section explained a method to determine whether a vehicle can move from one PCR to another by making a smooth turn, i.e. the connectivity between two PCRs. We then build a connectivity graph and search for a path which satisfies the maximum curvature constraint.

6.4.1 Connectivity Graph

Nodes in the connectivity graph represent the straight line segments within PCRs. An end point of such a line segment is either a candidate corner inside an overlap with another PCR or the initial or goal position of a point robot. An edge from a node N_i to N_j exists if and only if the corresponding line segments L_i and L_j share an end point and there is a smooth turn from L_i to L_j as explained in Section 6.3. The cost (length) of the edge is the length of the partial path (a cubic spiral or a line segment)

1. from the start point of L_i to the midpoint of L_j, if the start point of L_i is the initial position,

2. from the mid point of L_i to the end point of L_j, if the end point of L_j is the goal position,

3. from the midpoint of L_i to the midpoint of L_j, elsewhere.

Let k be the maximum number of candidate turning corners in each overlap of PCRs. An upper bound on the number of nodes in the graph: $O(k^2n^{12})$ can easily obtained, because there are at most $O(n^4)$ PCRs and a sequence of three PCRs (there are $O(n^{12})$ of them) generates k^2 nodes in the connectivity graph. Considering that k does not increase with n in practice, we drop k^2 from the complexity and get $O(n^{12})$ as the total number of nodes.

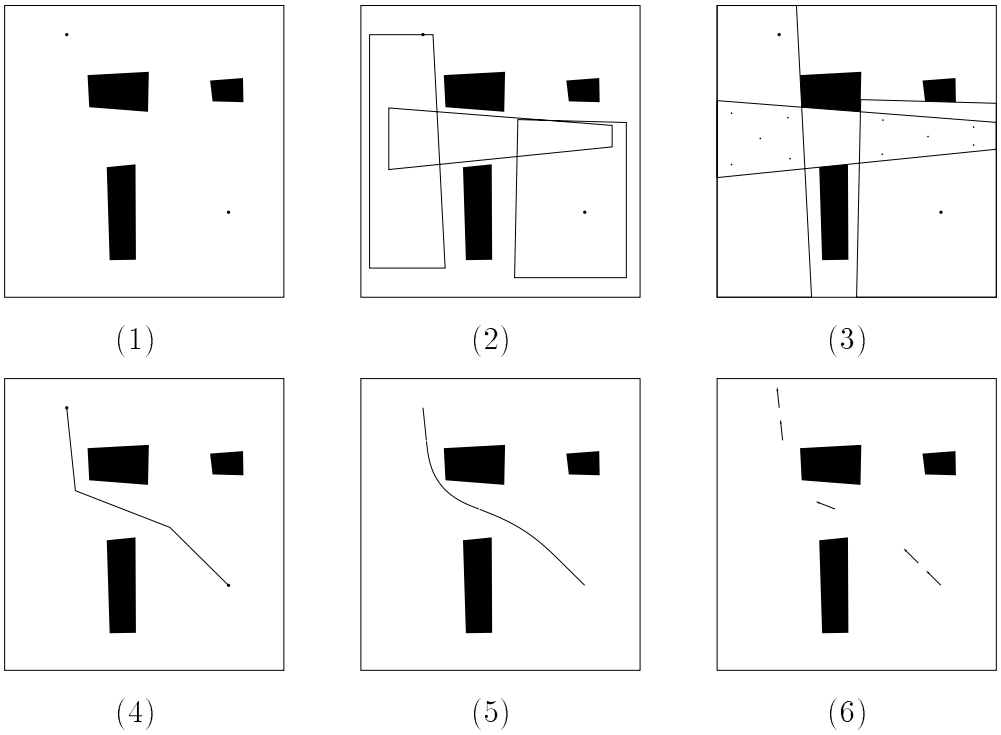
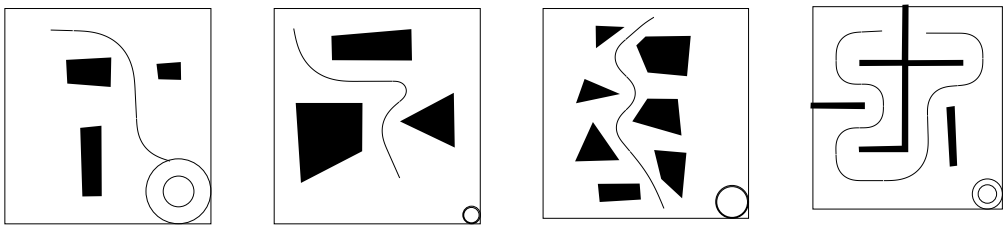


Figure 6.6: Steps Involved in Path Planning. **(1)** Initial and goal position is given. **(2)** Identify PCRs. (only those on the solution path are shown.) **(3)** Identify candidate turning points in overlap regions. **(4)** Find least cost path in connectivity graph, consistent with maximum curvature constraint. **(5)** Create smooth path by inserting cubic spirals. **(6)** Identify subgoals as start/end points of turns of the path.

6.4.2 A^* Search

We use the A^* algorithm (see [Nilsson 80]) to find a path in the connectivity graph. As a heuristic function, we use Euclidean distance from a current node (midpoint of its line segment) to a goal position.

Figure 6.6 shows the steps involved in the path planning. Figure 6.7 are the paths found. The paths are natural in the sense that large turns are taken where there is much space and smaller turns are taken where there is not much space.



(1) (2) (3) (4)

Figure 6.7: Paths Found. Inner circles at the bottom right have the maximum curvature given for searches, and outer circles have maximum curvature for the paths found (radius is the inverse of curvature). In these examples, only the centers of gravity of overlaps are used as candidate turning corners, which cause an odd detour in the example (2). This is improved by putting more candidate turning corners (see Fig. 6.9 (2-c)).

6.4.3 Complexity

We summarize the complexity of our algorithm as follows. Let N be the total number of nodes in the connectivity graph searched by the A^* algorithm. The complexity of the A^* search algorithm has been analyzed in [Martelli 77], and it was shown that

- The A^* algorithm requires $O(2^N)$ steps in the worst case if the so called *consistency assumption* does not hold for the heuristic function used.
- Martelli's algorithm (Algorithm B) has $O(N^2)$ running time and never requires more steps than the algorithm A^*.

Our heuristic function (Euclidean distance) satisfies the consistency assumption, and both A^* and B will have the same behavior. We can always replace A^* with algorithm B, and obtain an upper bound for the complexity: $O(N^2) = O(n^{24})$ where n is the total number of obstacle walls, because N is at most $O(n^{12})$.

Rueb and Wong have also reported an $O(n)$ performance result for their experiment as opposed to the $O(n^4)$ upper bound (see Section 6.2.3). This immediately makes our algorithm run in $O(n^6)$ time instead of $O(n^{24})$. Furthermore, average case running time for A^* search is much better because of the heuristic associated.

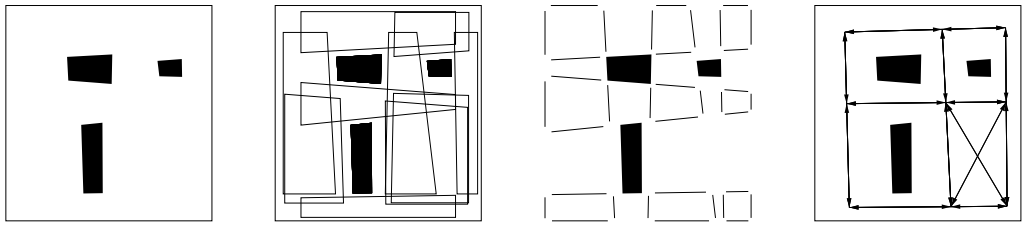
6.4.4 Experimental Results

Judging from our experiments, the algorithm can find a path efficiently when space is relatively uncluttered or the maximum curvature given is large. In these cases we do not need to put multiple candidate turning corners in the overlap regions. When the space is cluttered or the maximum curvature given is small, the algorithm can still find a path by adding new candidate turning corners. However this makes graph search much slower.

To remedy this problem, we used *road map* distance as another heuristic function. Road maps are explained in [Rueb and Wong 87], as an efficient way to find the collision free polygonal path. Road maps can be built quite easily using *maximal overlapping regions* (see figure 6.8). Although this is not an admissible heuristic function[2], it can shorten the search time considerably when Euclidean distance does not provide a good cost estimation (for example, when detours are necessary).

Figure 6.9 are the paths found given various conditions for the best first search. Table 6.1 shows the times needed to find the paths in the figure. It is possible to further improve the search time when we use the road map

[2]This is because a polygonal path found using a road map is not generally the shortest path.



(1) (2) (3) (4)

Figure 6.8: Maximal Overlapping Regions and Road Map. **(1)** Obstacles. **(2)** PCRs. **(3)** Maximal overlapping regions. **(4)** Road map. A road map is constructed by connecting two maximal overlapping regions contained in a PCR.

distance as a heuristic function, by taking into account the minimum width of a polygonal path. The wider a polygonal path is, the better chance we have to find a smooth path along the polygonal path.

6.5 Summary

An algorithm to find a smooth path with maximum curvature constraint has been presented. The algorithm can find a path efficiently when the space is relatively uncluttered. When the space is more cluttered, the algorithm can still find a path by adding new candidate turning points. The obtained smooth path is natural in the sense that it can take large turns where there is much space and smaller turns where there is not much space.

After the path planning is finished, the globally optimum path obtained can be converted to a sequence of subgoals (a pair of position and orientation, identifying the start and end points of a turn), and then passed to a local path following module (such as the one referred in [Kanayama and Hartman 89, Nelson 89]). Because of the smoothness of the path, it will be much easier to follow the path.

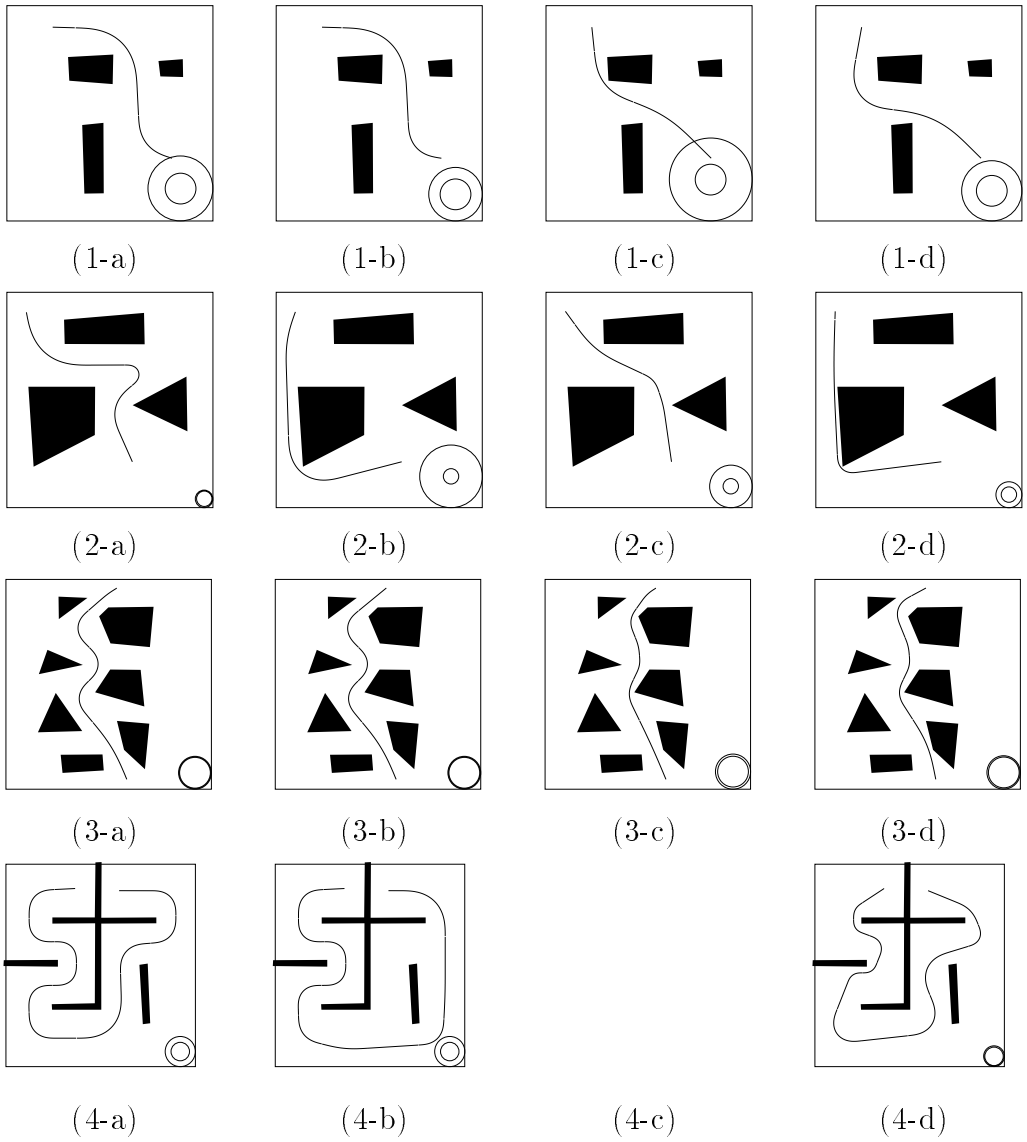


Figure 6.9: Smooth Paths Found for 4 Environments. Conditions for the search are **(a)/(b)** a single turning point using Euclidean/road-map distance, **(c)/(d)** multiple turning points using Euclidean/road-map distance. We could not obtain a solution for (4-c) in a reasonable amount of time because of the long detour necessary. The example seems to show the worst case for A^* search, since Euclidean distance is not a good heuristic at all in the example. Inner circles at the bottom right of the examples have the maximum curvature given for searches, and outer circles have maximum curvature for the paths found.

example	candidate corners	heuristic function	$1/\kappa_{max}$ given	$1/\kappa_{max}$ obtained	path length	search time (sec)
1-a	single	h_1	50	106	674	21
b		h_2		87	679	9
c	multiple	h_1		135	614	393
d		h_2		98	678	136
2-a	single	h_1	25	29	799	11
b		h_2		102	874	5
c	multiple	h_1		70	640	90
d		h_2		42	839	117
3-a	single	h_1	50	53	758	119
b		h_2		53	757	75
c	multiple	h_1		57	684	2300
d		h_2		54	705	283
4-a	single	h_1	30	49	1763	18
b		h_2		49	1855	7
c	multiple	h_1		-	-	-
d		h_2		34	1605	62

Table 6.1: Search Time and Path Length for the examples shown in Fig. 6.9. Single/multiple denotes a single/multiple turning point(s) for each overlaps. h_1 uses Euclidean distance, while h_2 uses road-map distance. Search time was measured on a Symbolics 3670 without a floating point arithmetic hardware.

6.6 Related Work

The complexity of the continuous curvature path planning problem with the maximum curvature constraint is not known. An easier problem, the path planning problem with the maximum curvature constraint, has been studied recently. In the problem, the curvature continuity of the path is not required, although the path is required to be in C^1 class.

[Fortune and Wilfong 88] presents an exponential time procedure to decide whether such a path exist given start and end position/orientation. Jacob and Canny have developed a polynomial algorithm to find a ϵ-approximation for the shortest path [Jacobs and Canny 89]. Their algorithm is based on Dubins' theorem [Dubins 57] which says that the shortest path in open space with the maximum curvature constraint consists of at most three pieces, each of which is either a straight line segment or an arc of a circle of radius equal to the inverse of the maximum curvature.

It may be possible to modify a path obtained using Jacob and Canny's algorithm so that the modified path has continuous curvature. However, we did not choose to do so, because of the following reasons.

- It has been claimed that paths found using free space decompositions are better suited for navigational purposes, since the convex regions can be used to accurately position a robot locally.

- Although the worst case complexity of our algorithm is not favorable compared to Jacob and Canny's, our algorithm runs fast in less cluttered environments because of the use of A^* search with good heuristics.

- We are interested in applying the smooth path planning for mobile robots to the path planning for manipulators, so that a 3-D extension of a smooth

path planning algorithm is needed. Dubins' theorem on which Jacob and Canny's algorithm is based holds only in 2-D, which makes it difficult to extend their algorithm to 3-D. However, our algorithm which is based on the PCR decomposition of free space can be extended to 3-D as we will see in the next chapter.

Chapter 7

Path Planning for the Continuous Manipulator

In this chapter, we plan paths for the continuous manipulator model. First, we show that once a smooth path is found for a point robot, the motion schemas are determined for the continuous manipulator model to extend itself through subgoals along a smooth path until its tip reaches the goal. By executing the schemas, we obtain a path for the continuous manipulator.

To find smooth paths in 2-D, we use the algorithm presented in the previous chapter. The algorithm was based on decomposing free space into primary convex regions. To find smooth paths in 3-D, we decompose 3-D free space into primary convex regions as well.

7.1 Achieving Subgoals along a Smooth Path

In Section 5.2, we demonstrated how to fold the manipulator and then extend it to reach a goal *if subgoals are given*. In Chapter 6, we have presented an algorithm to find a smooth path and subgoals for a *point robot*. By combining the two results, we can obtain a path for a continuous-curvature manipulator as follows.

7.1.1 Finding Subgoals

The algorithm to find a smooth path for a point robot is easily modified to find subgoals through which the continuous manipulator is extended.

We assume there is enough open space around the base to fold the manipulator. First, locate the folded manipulator. The primary convex region which contains the folded manipulator is called the *base PCR*. Since the manipulator is folded as a circular arc, we can extend the manipulator from anywhere on the circle by rotating it around the base. Hence, as initial states of the graph search, we use tangent lines to the circle from all candidate turning corners in the overlaps with the base PCR. These initial states correspond to partial paths through which we can extend the manipulator.

After defining the initial states, graph search proceeds exactly in the same manner for finding a smooth path for a point robot. And subgoals for motion schemas are obtained as start/end points of turns of the path (figure 7.1). The modification does not change the complexity of the original algorithm to find smooth paths.

7.1.2 Achieving Subgoals

To change from a current configuration to a target configuration represented by the subgoals in figure 7.1, we use a *retract, rotate, and extend* strategy (figure 7.2).

The $O(n^{24})$ upper bound for the complexity of the algorithm does not increase by executing the motion schemas, if we allocate a unit cost to each schema execution. This is because the number of segments is equal to the number of subgoals, which is at most the twice of the number of PCRs ($O(n^4)$). Iterations associated with the schema execution are for fine tuning the five curvature parameters to achieve subgoals precisely, and they converge rapidly. Also refer to Section 5.3.1 on this argument.

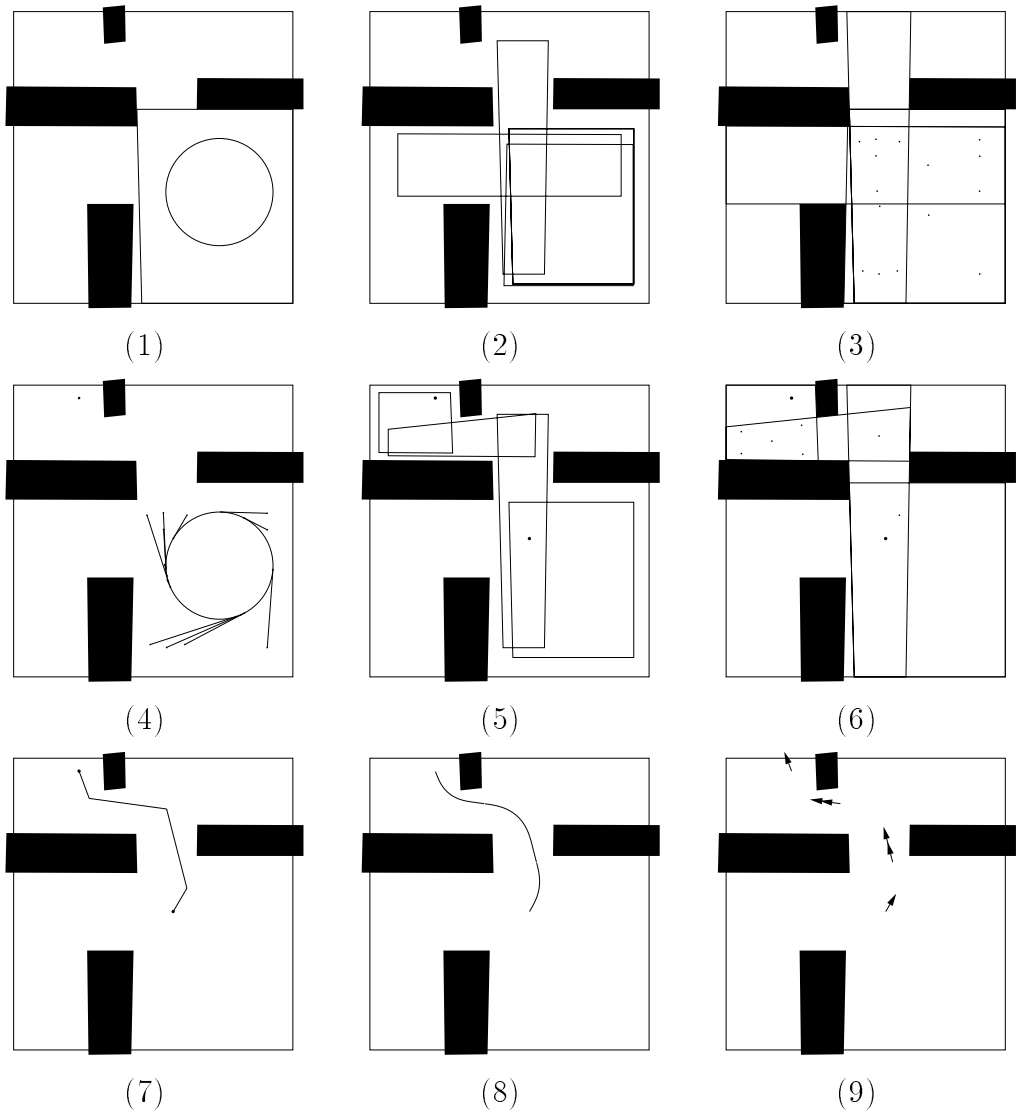


Figure 7.1: Path Planning for Manipulator. **(1)** Base PCR. **(2)** Identify PCRs which overlaps with base PCR. **(3)** Identify candidate turning points in base PCR. **(4)** Select initial directions to all candidate turning points in base PCR. **(5-9)** Same as Fig. 6.6

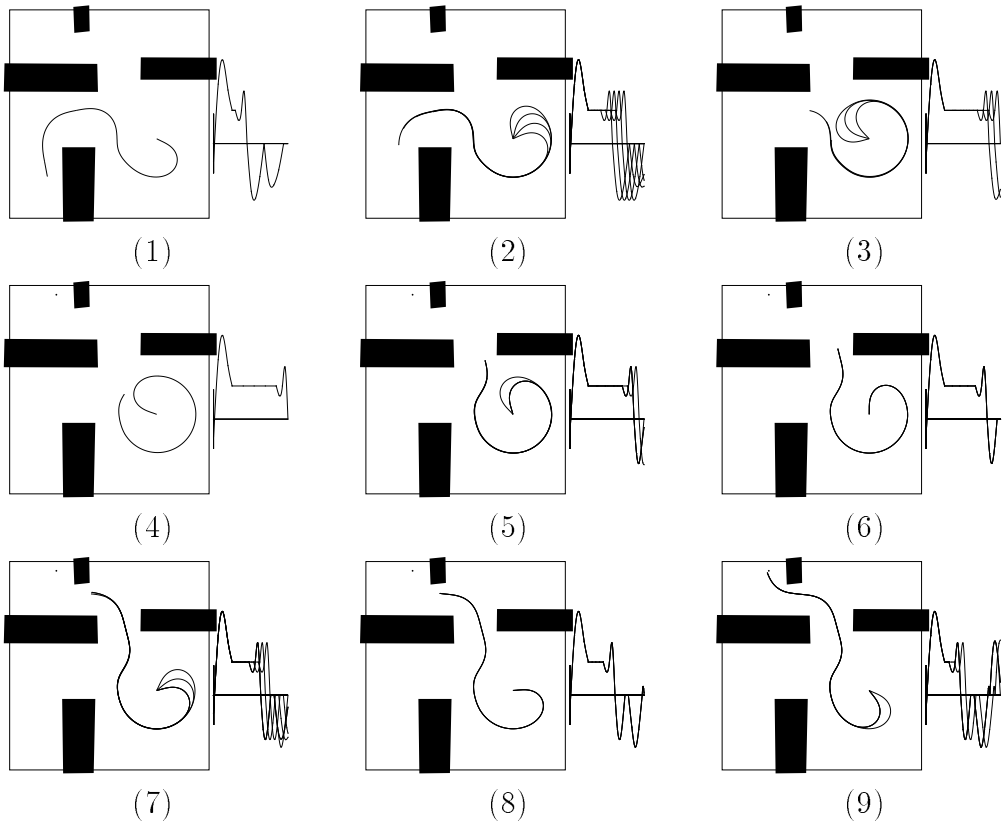


Figure 7.2: Retract, Rotate, and Extend: We use the same decompositions and schemas shown in figure 5.4. For additional subgoals, we add more segments. Cubic spiral trajectories are selected for feed/retract schemas.

7.1.3 Comment on the Meaning of Convexity

Note that the convexity concept for manipulators generally refers to convexity in the configuration space, i.e. high dimensional space of joint variables. However, the convexity concept in the task space (2-D and 3-D) becomes meaningful by restricting the trajectories of manipulators to those along smooth paths in the task space.

As far as highly redundant manipulators are concerned, restricting the class of trajectories is a reasonable constraint.

- There should be a large number of paths for such manipulators, and we are looking for one of them.
- Even with the constraint, we would not miss a path if one exist for a completely flexible manipulator with infinitely many DOF.
- The constraint becomes less restrictive with the number of DOF in the manipulator. As we will see in the next chapter, the maximum curvature constraint for smooth paths improves with the number of DOF.
- We need to build a configuration space map in order to explore all the possible trajectories. But this is practically impossible for highly redundant manipulators.

We will revisit the issue in Section 9.2.

7.2 Extend to 3-D

For highly redundant manipulators, two approaches are feasible for path-planning in 3-D space. Brooks used a $2+\frac{1}{2}$-D approach for pick-and-place

operations, in which 3-D space is treated as a stack of 2-D spaces. Alternatively, we could decompose free space into convex polyhedra and use continuous 3-D curves with torsion to move from one polyhedron to another.

7.2.1 $2 + \frac{1}{2}$-D Approach

Brooks proposed decomposing free space into generalized cones in order to find a path for mobile robots [Brooks 83a]. He then used the same free space representation to plan a collision free path for manipulators [Brooks 83b]. With this approach, free space in 3-D is represented by horizontal 2-D slices. The *free space skyward property* is assumed: if a point is in free space, then all the points above it are also in free space. As a consequence of the free space representation, hand movement of the manipulator is restricted to 4 DOF, 3 DOF for planar movements (two translations and one rotation along a vertical axis) in horizontal planes, and 1DOF for vertical translations.

We can take a similar approach with our free space representation using primary convex regions. We restrict the movements of each part of a manipulator within a plane, which are then combined to produce 3-D movements for the whole manipulator. For example, in order to move a *hand* part of the manipulator vertically from one horizontal plane to another, planning for an *arm* part in a vertical plane will suffice. In general, the decomposition of a 3-D planning problem into a set of 2-D problems would become easier with more DOF in a manipulator, because each part of the manipulator becomes more independent. When this decomposition is not the possible, we decompose 3-D free space directly (see Section 7.3).

7.2.2 About Hypergraph Method

Now let us turn to the subject of 3-D free space decomposition. In chapter 6, we used the hypergraph method [Rueb and Wong 87] to find the primary convex regions in 2-D. Unfortunately, the hypergraph method can not be extended directly to 3-D. In 2-D, walls segments of primary convex regions make certain kind of cycles in a graphical representation of wall segments. The hypergraph method finds primary convex regons by searching for those cycles. But this property of primary convex regions does not hold in 3-D.

In the study of computational geometry, it is well know that problems in 3-D are often considerably harder than the corresponding 2-D.

For example, there is an $O(n^2)$ efficient and optimal algorithm called *visibility graph method* to find the shortest path in 2-D space with polygonal obstacles. However, the shortest path problem in 3-D space with polyhedral obstacles is much harder. The difficulty arises from the fact that the only constraint on the vertices of the shortest path is that they line on the edges of the polyhedral obstacles. In this sense, the problem is not discrete. In [Sharir and Schorr 84], a doubly exponential procedure is presented for solving the discrete subproblem of determining the sequence of obstacle edges through which the shortest path passes.

The projection method [Leven and Sharir 87] leads to an $O(n^2 logn)$ algorithm for a line segment moving in 2-D space. The retraction method [O'Dunlaing and Yap 85] leads to an $O(n \log n)$ algorithm for a disc moving in 2-D space. However, these efficient algorithms do not extend to 3-D, and it is said that the problem of obtaining efficient path planning algorithms for specific robot systems in 3-D are mostly open [Yap 87].

7.3 3-D free space decomposition

Although we cannot extend the hypergraph method to 3-D, there is another method which can ealily developed to 3-D. The method was developed in [Singh and Wagh 87] to find primary convex regions efficiently in 2-D for a restricted class of obstacle layout.

We first explain Singh and Wagh's decomposition in 2-D, then extend it to 3-D. We also discuss a free space paritioning method at the end of the section.

7.3.1 Singh and Wang's method to find Primary Convex Regions

The algorithm assumes that obstacles are approximated by iso-oriented rectangles in which the edges are parallel to the coordinate axes. Primary convex regions in the obstacle layout are the largest rectangular free areas. Given a map of boundaries and n obstacles, the environment is partitioned by the edges into a grid of at most $(2n+1) \times (2n+1)$ rectangles where n is the number of obstacles. Each partition in the grid can be represented by a pair of binary strings of length $2n+1$, the first string representing x position, the second y position. For example, a partition which is second from the left and third from the bottom in Fig. 7.3 can be represented

010 001

The notation can be used to represent a larger rectangular area which consists of several partitions. The pair of strings

011 110

represents a union (binary OR) of four partitions:

010 100

010 010

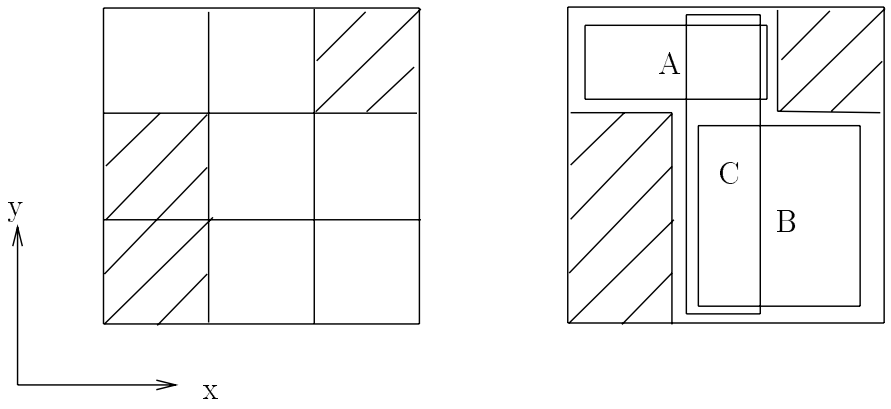


Figure 7.3: Fuse Free Regions in 2-D

001 100

001 010

The following algorithm identifies the largest free rectangular regions by fusing the free partitions recursively.

1. Represent each horizontal strip.
2. Find all the continuous horizontal strips.
3. Make a list of strings generated by Step 2 in such a way that
   - strings are grouped by the value of y portion (second sub-string).
   - groups are ordered according to the value of y portion.
4. Generate a new list of strings from the old list using the following rules
   - The ith group in a new list is generated by combining each string from the ith group and $i+1$th group from the old list.

- Two strings are combined by taking logical OR of Y portions and by taking logical AND of x portions. If x portion of the new string is all zero, discard the string.
- Each time a new string is generated, check off all elements in the generating groups in the old list which are covered by the new string. A string S_1 is covered by a string S_2 when $OR(S_1, S_2) = S_2$.

5. Repeat the previous step while two or more groups are generated.
6. The strings which have not been checked off when the algorithm terminates represent primary convex regions.

Example

When we apply the procedure to an example in Fig. 7.3, we obtain the following.

List 1

group 1: 011 100 $\surd$

group 2: 011 010 $\surd$

group 1: 110 001 A

List 2

group 1: 011 110 B

group 2: 010 011 $\surd$

List 3

group 2: 010 111 C

A, B, and C are the primary convex regions obtained (Fig. A-B-C). When we notice the y portions of strings in the lists, we can see that free regions are fused to x direction in Step 2 (list 1), and then fused to y direction recursively by building list 2 and 3.

Let n be the total number of obstacles. There are at most $O(n^2)$ primary convex regions, because each primary convex region is bounded by two pairs of obstacle edges (in x and y directions), and for at least one of the two pairs, it is the only primary convex region bounded by the pair. Note that there are $O(n^2)$ obstacle edge pairs in each direction.

[Singh and Wagh 87] did not report the complexity of the algorithm, but a loose upper bound $O(n^5)$ can easily be obtained where n is the total number of obstacles, because

- There are at most $O(n)$ lists (N_A).
- Each list contains at most $O(n)$ groups (N_B).
- Each group contains at most $O(n)$ strings (N_C).
- Each string is $O(n)$ long (N_D).

The cost of generating a new string is bounded by

$$(N_A * N_B * (N_C * N_C) * N_D = O(n^5)$$

and checking off the covered string is also bounded by

$$(N_A * N_B * N_C) * (N_C + N_C) * N_D = O(n^5)$$

7.3.2 Finding Primary Convex Regions in 3-D

Let us extend Singh and Wagh's algorithm to 3-D. Now, each string represents a rectangular parallelepiped and consists of three sub-strings: x,y, and z portions. In the 2-D algorithm, fusing free rectangles in the x direction is the first step, then the recursive procedure is applied for fusing in the y

direction. In the 3-D case, after free rectangular parallelepipeds are fused in x direction, there are still two directions (y and z) in which they can be fused (see Fig. 7.4).

Hence the 2-D procedure should be modified so that

- To fuse in the y direction and generate a new string, OR the y portion of the strings in the old groups, and AND the x and z portions of the strings.
- To fuse in the z direction and generate a new string, OR the z portion of the strings in the old groups, and AND the x and y portions of the strings.
- The recursive procedure to fuse in both y and z directions should be ordered properly to make the procedure systematic and efficient. For this reason, we expand the string, group, and list hierarchy by collecting list into *levels*. We order new element generation so as to produce each levels sequentially.
- A group of free regions to be fused in both y and z directions must be ordered in two ways: according to their y values and to their z values.

Fig. 7.6 shows how a modified procedure works to identify all the largest free rectangular parallelepiped for an example in Fig. 7.5.

There are at most $O(n^4)$ primary convex regions, because each primary convex region is bounded by three pairs of obstacle edges (in x,y,and z directions), and for at least two of the three pairs, it is the only primary convex region bounded by the two pairs. The total number of two pairs of obstacle edges is $O(n^4)$, since there are $O(n^2)$ pairs in each direction.

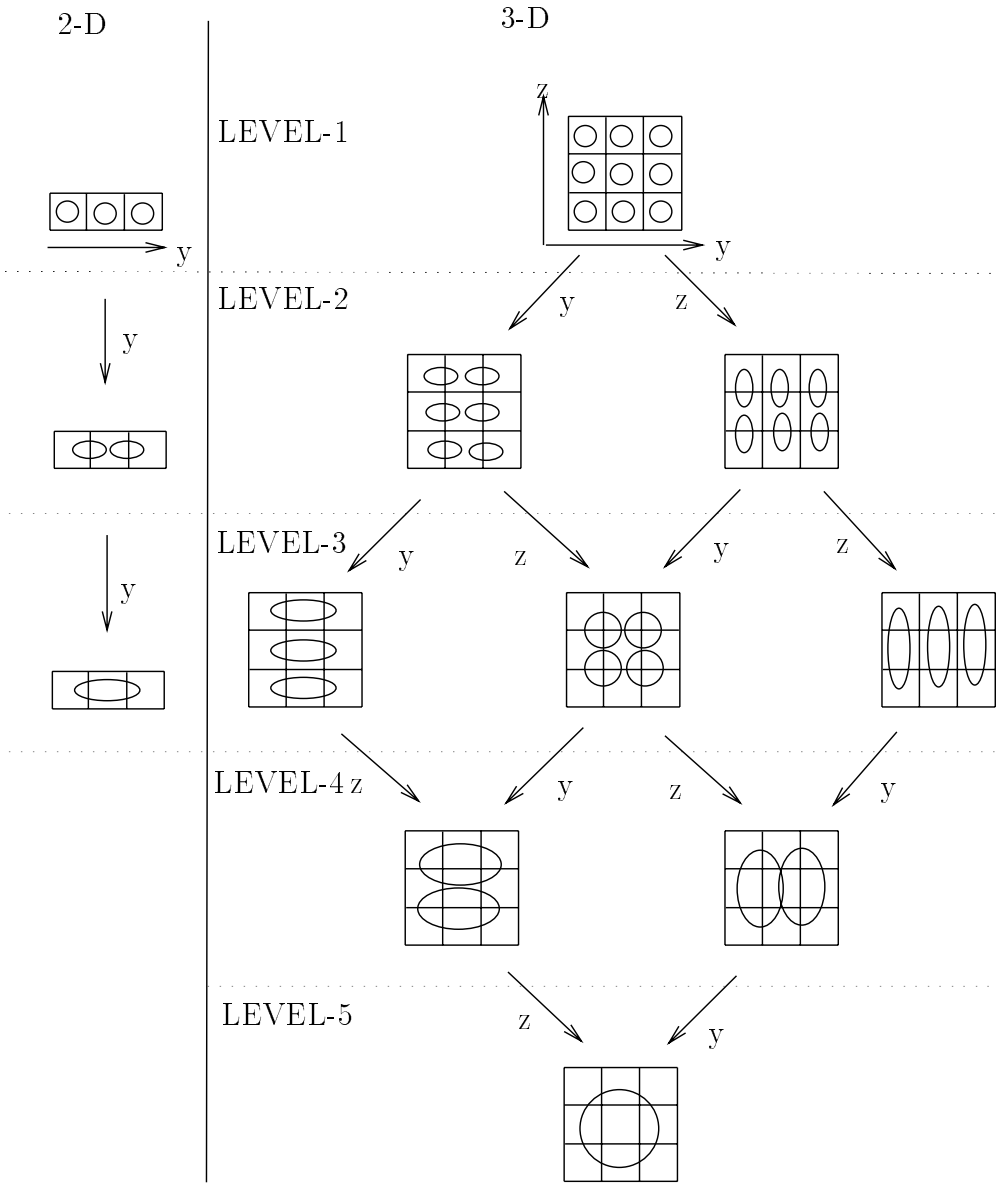


Figure 7.4: Fuse Free Regions in 2-D and 3-D

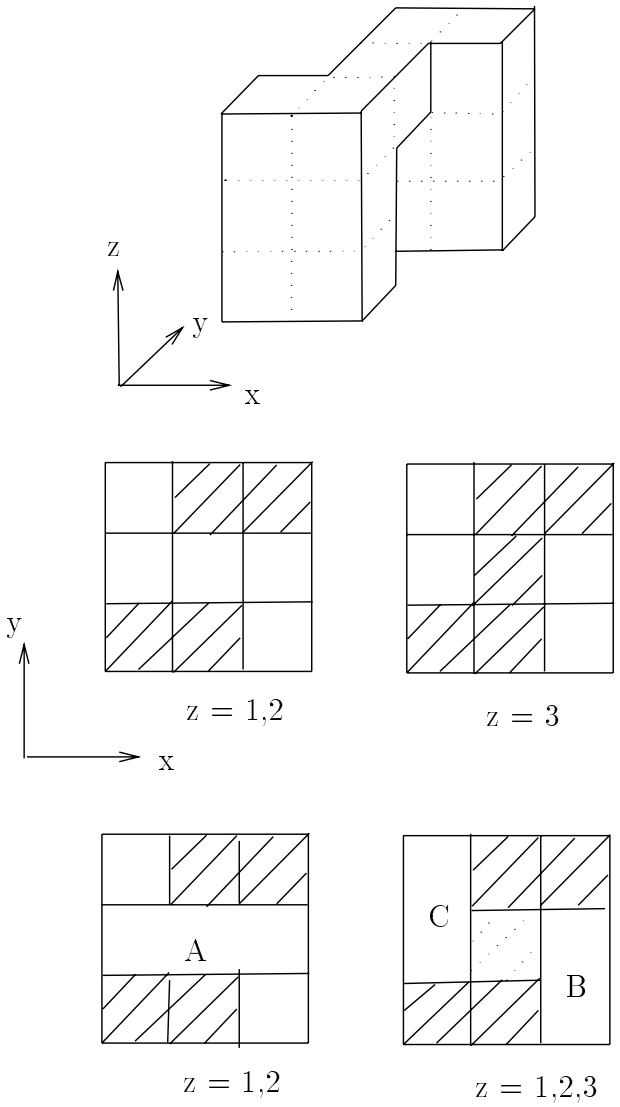


Figure 7.5: Arch Example

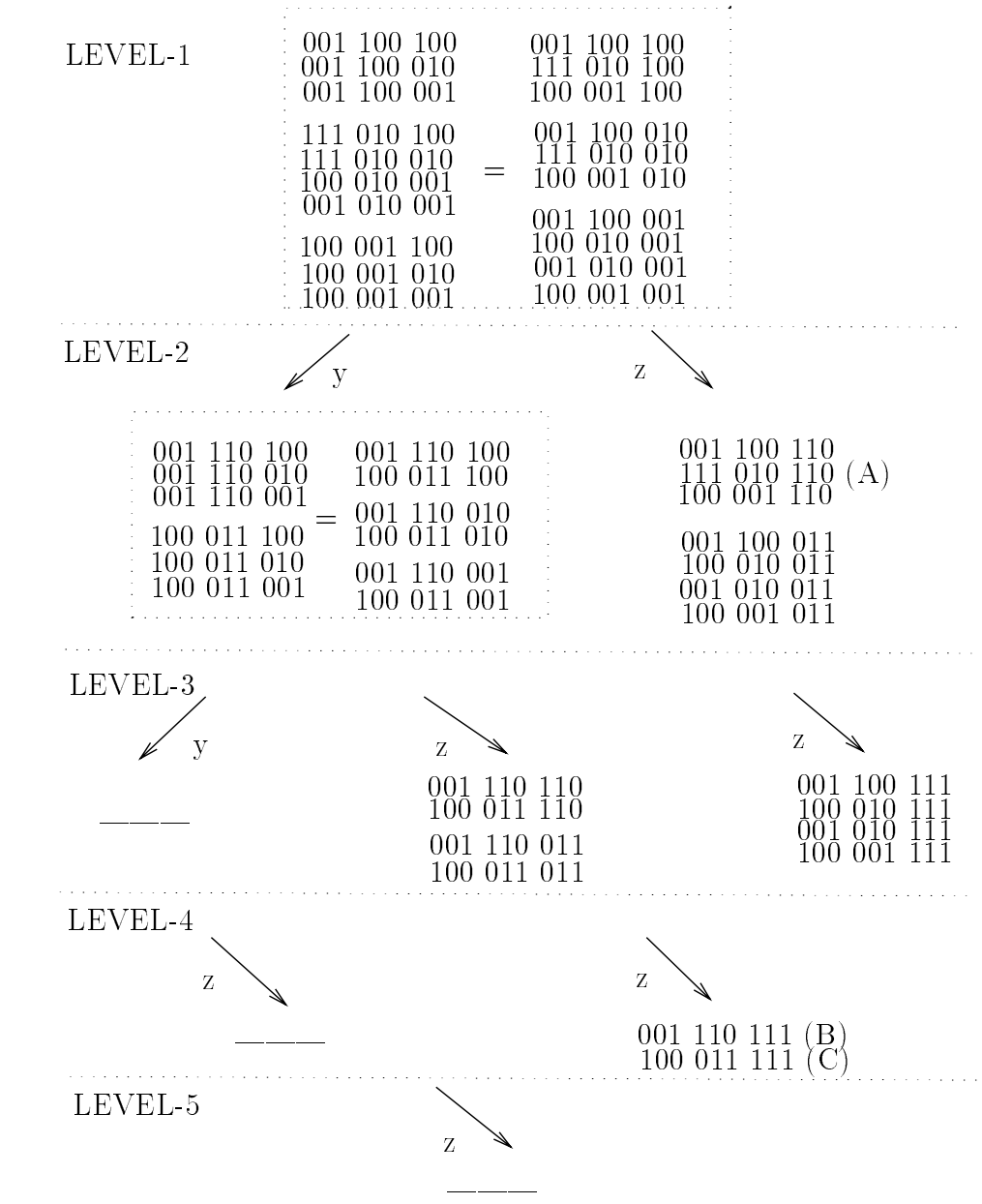


Figure 7.6: Finding Primary Convex Regions for the Arch Example

A loose upper bound for the complexity of the extended algorithm for 3-D, $O(n^6)$, can be obtained as follows.

- There are at most $O(n)$ levels (N_{level}).
- There are at most $O(n)$ lists in a level (N_A).
- Each list contains at most $O(n)$ groups (N_B).
- Each group contains at most $O(n)$ strings (N_C).
- Each string is $O(n)$ long (N_D).

The cost of generating a new string is bounded by

$$(N_{level} * N_A * N_B * (N_C * N_C) * N_D = O(n^6)$$

and checking off the covered string is also bounded by

$$(N_{level} * N_A * N_B * N_C) * (N_C + N_C) * N_D = O(n^6)$$

7.3.3 Free Space Partitioning Methods

Path planning based on free space partitioning into disjoint convex regions has also been studied. [Arkin 89] proposed the use of a *meadow map* for path planning. In this scheme, free space is decomposed into disjoint convex regions. Polygonal paths are found using A^* search in the connectivity graph of convex regions. Candidate turning corners between the two regions are put either at the midpoint on the boundary, or at three points on the boundary. After a path is found, he tries to optimize the path by removing unnecessary corners.

[Rao and Arkin 90] extends the meadow map method to 3-D space for use by a flying robot. The map is now called *crystal map*, where free space is decomposed into disjoint convex polyhedra. Since the boundary between two convex polyhedra is now a polygon, candidate turning corners are put either at the center of gravity of a polygon or at multiple points on the polygon.

Compared to the our method based on free space decomposition into primary convex regions, these methods generally lead to more regions generated, more search for paths, and more work for path optimization. But they have the advantage of ease of extension into 3-D space.

Free space decomposition for path planning is a variation of the convex decomposition problem which has been extensively studied in computational geometry [Lee and Lee 90, Ronse 89]. While most of the results are for 2-D, [Chazelle 84] presents an $O(nN^3)$ algorithm which partitions a polyhedra in 3-D into $O(N^2)$ convex parts where n denotes the complexity of the polyhedra and N denotes the number of *reflex angles* (those that are $\geq \pi$).

However, applicability of Chazell's algorithm to path planning has yet to be investigated. The motivation for the free space decomposition methods is to find a decomposition suitable for path planning and navigation of mobile robots. Many algorithms exist for convex decomposition of polygons, but they do not necessarily satisfy the requirement for path planning and navigation.

Chapter 8

Mapping the Solution to a Jointed Arm

In this chapter, we consider the mapping from the continuous manipulator model to a highly redundant jointed arm. We first introduce a straight forward mapping which we call the every-other-joint mapping. Then, the mapping error for the every-other-joint mapping is evaluated for a class of continuous configurations in terms of maximum curvature. A problem of finding a better mapping will be discussed to improve the approximation by a jointed arm. Finally, we briefly introduce a dynamic simulation which we performed after the solution had been mapped to a jointed arm.

8.1 Every-Other-Joint Mapping

We provide a mapping to a jointed arm which has an even number of links of the same length. First, group links into pairs of consecutive links. Then, place odd numbered joints $(1, 3, \ldots)$ on the continuous solution in such a way that they are equi-distant on the curve. The positions of the even numbered joints $(2, 4, \ldots)$ are automatically determined in the process. In fact, there are two positions left for the joint, and we choose one which is closer to the continuous configuration. (See Fig. 8.1). Using this mapping, the trajectory for the continuous manipulator in Fig. 7.2 is mapped to a trajectory for an arm with 12 joints in Fig. 8.2.

Joint values can easily be obtained once the mapping has been finished. Let x_i,y_i be the coordinates for the i-th joint $(i = 1, 2, \ldots)$. Then, if we

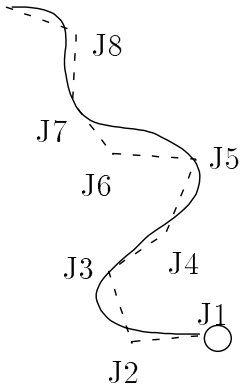


Figure 8.1: Every-other-joint Mapping to a Jointed Arm

use the notation ϕ_i as in Fig. 8.3,

$$\begin{aligned} \theta_i &= \arctan \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \\ \phi_1 &= \theta_1 \\ \phi_i &= \theta_i - \theta_{i-1} \quad (i \geq 2) \end{aligned}$$

Fig. 8.4 is a graph of joint rotations for the trajectory in Fig. 8.2. The first joint (Joint 1) is the base. The values for all joints except the first joint are small. Their absolute values are less than 90 degrees. Joints 2 through 6 have almost constant values, because they are kept folded. Joints 7 through 12 correspond to the part of the manipulator retracted and extended. Fig. 8.5 is a graph for these joints (7 through 12). Two modes of movement are apparent in the graph: joints 8 through 12 follow the tail (Joint 7) while retracting, and joints 11 through 7 follow the tip (Joint 12) while extending.

8.2 Evaluating Mapping Errors

In Chapter 6, we found smooth paths with maximum curvature constraint. If we can determine an upper bound on the mapping errors as a function

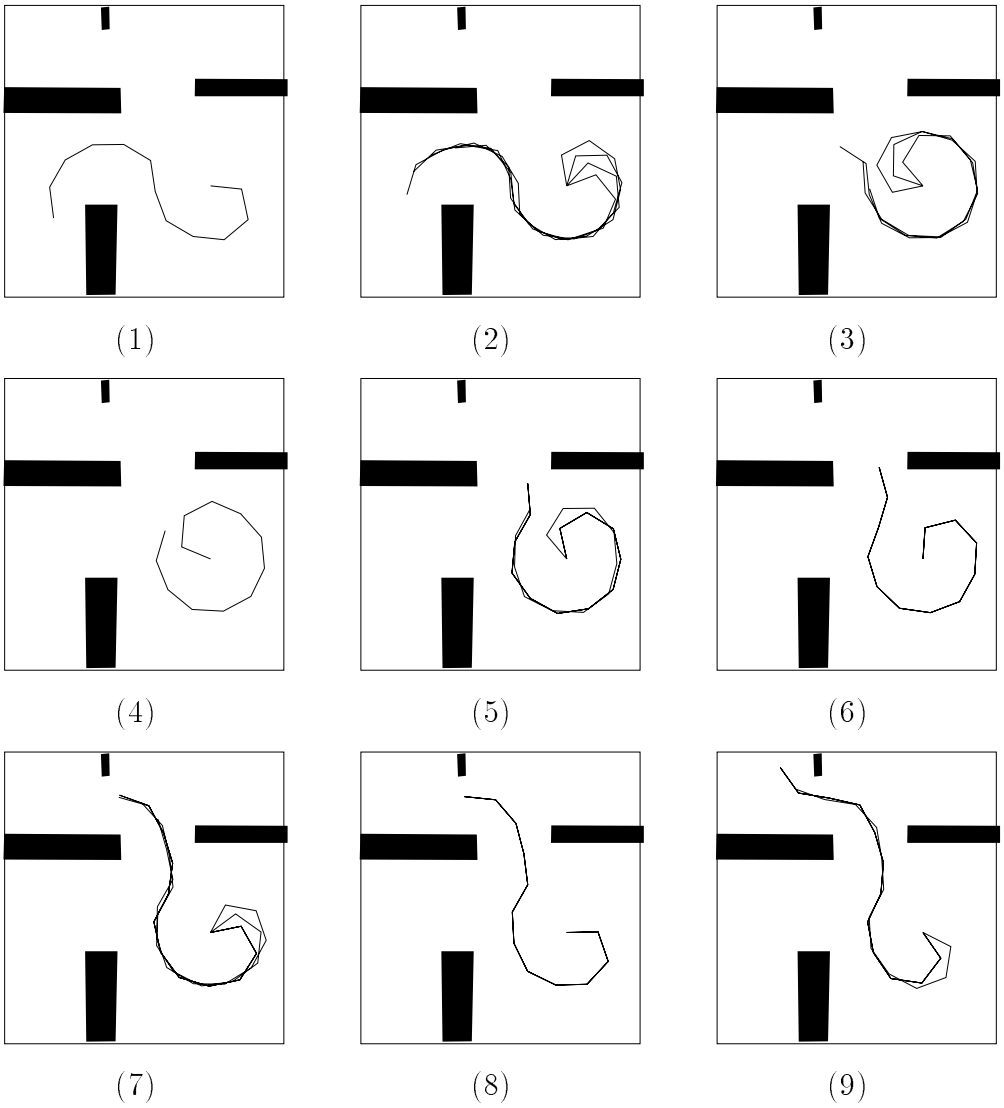


Figure 8.2: Jointed Arm Trajectory

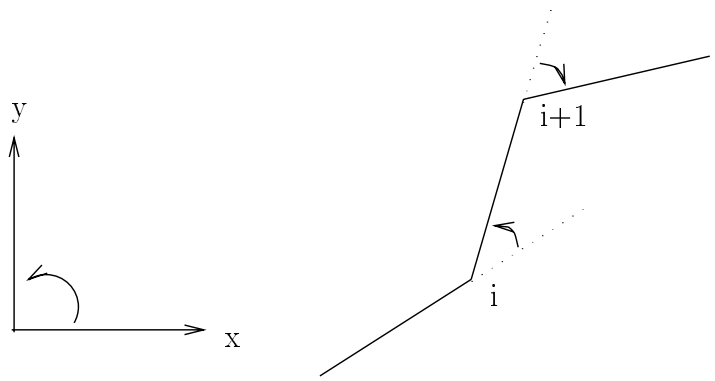


Figure 8.3: Joint Rotation

of the maximum curvature, we can "grow obstacles" by an appropriate amount in advance so that a collision free configuration for the continuous manipulator within the maximum curvature constraint is guaranteed to be mapped to a collision free configuration for a jointed arm.

It seems difficult to evaluate the mapping errors in general. However, it is possible to evaluate the errors if configurations of the continuous manipulator model consist only of straight lines and tangent cubic spirals. This is the class of configurations we have obtained using the algorithm in Chapter 6. Since the every-other-joint mapping is a local mapping scheme, only the mapping for two consecutive links has to be considered. Futhermore, if we assume the following:

> Each cubic spiral segment (including the straight line segments at both ends, if they exist) is longer than $2 * l$, where l is the length of each link of a jointed arm.

only two cases are left (Fig. 8.6) :

1. Single arc case: both ends of the link pair are on the same cubic spiral.

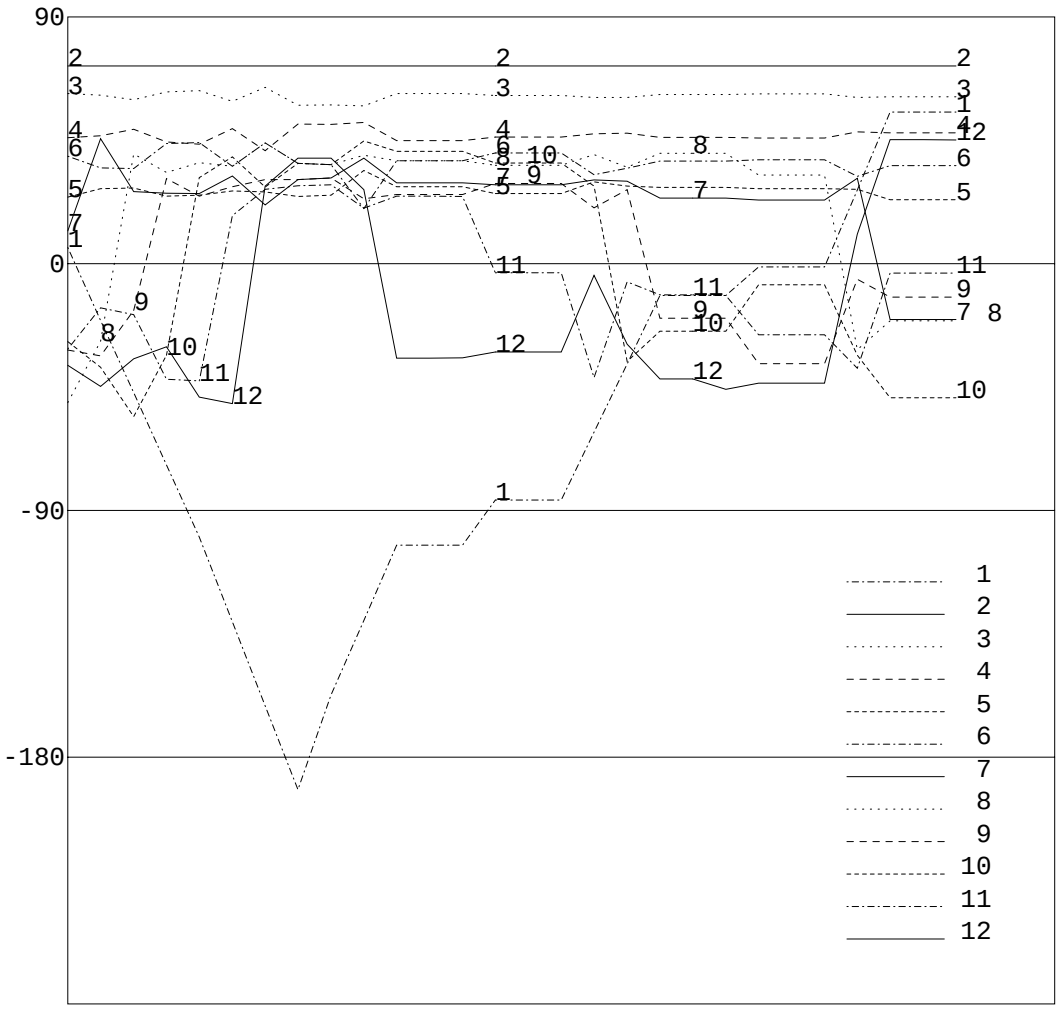


Figure 8.4: Joint Rotations for the Trajectory

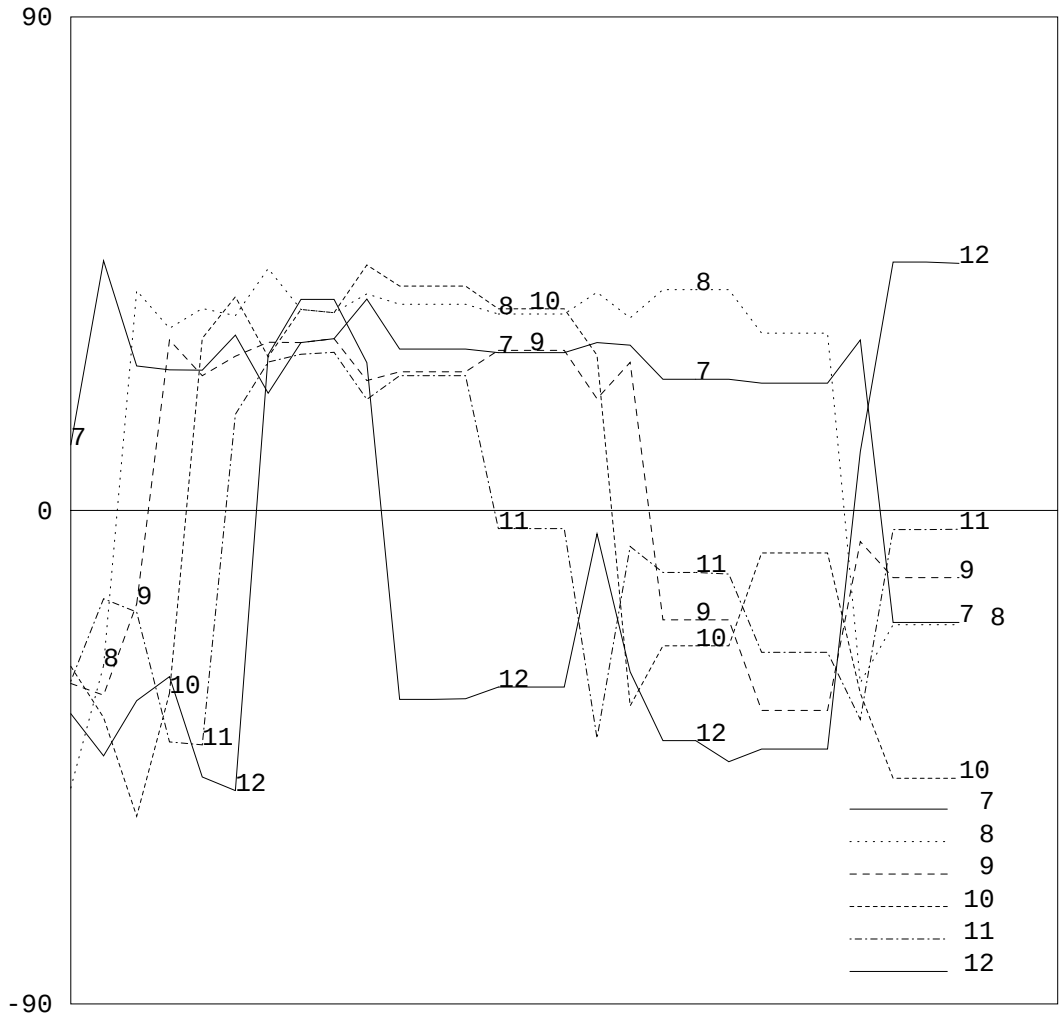


Figure 8.5: Joint Rotations for Tip Joints

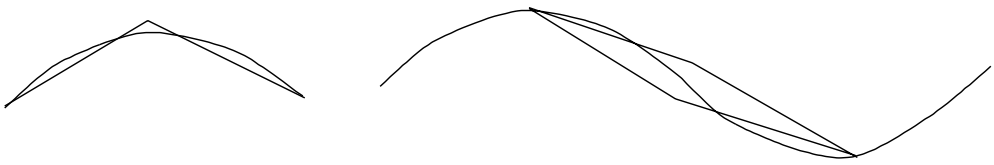


Figure 8.6: Single Arc Case and Tangent Arcs Case

2. Tangent arcs case: both ends are on consecutive cubic spirals with opposite sign of curvature. Tangent arcs with the same sign is similar to the single arc case and are less critical in terms of errors.

8.2.1 Single Arc Case

In order to evaluate the single arc case, we use a circular arc whose curvature is equal to the maximum curvature of the cubic spiral. This gives us an upper bound on the error (see Fig. 6.3 for comparison of circular arcs and cubic spirals). In the every-other-joint mapping , we have the following relation.

$$\begin{aligned} 2l &= 2r\theta \\ \theta &= \frac{l}{r} \end{aligned} \tag{8.1}$$

where

$$\begin{aligned} l &: \text{length of a link} \\ r &: \text{radius of the circular arc} \\ \theta &: \text{arc angle for a single link} \end{aligned}$$

Also see Fig. 8.7. We now assume the following.

$$\theta \leq \pi \tag{8.2}$$

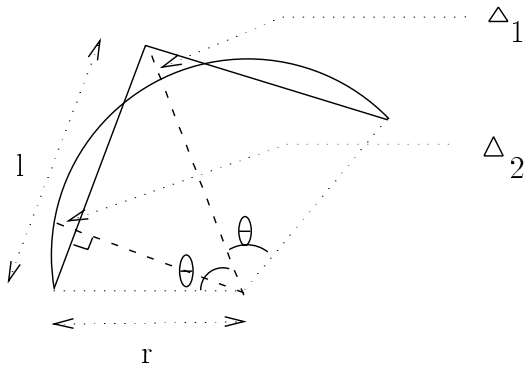


Figure 8.7: Single Arc Case with Δ_1 and Δ_2

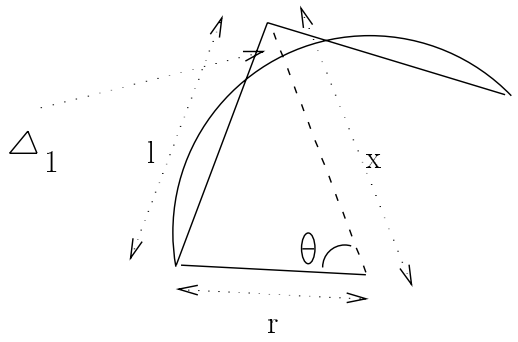


Figure 8.8: Δ_1 and x

It is apparent that the maximum mapping error is either Δ_1 or Δ_2 in the figure.

Let us try to express Δ_1 in terms of l and θ. Let x be the distance from the arc center to the second joint (see Fig. 8.8). Then

$$x = r + \Delta_1$$

From Cosine Theorem,

$$l^2 = x^2 + r^2 - 2xr\cos\theta$$

Therefore

$$x = r\cos\theta + \sqrt{r^2\cos^2\theta - (r^2 - l^2)}$$

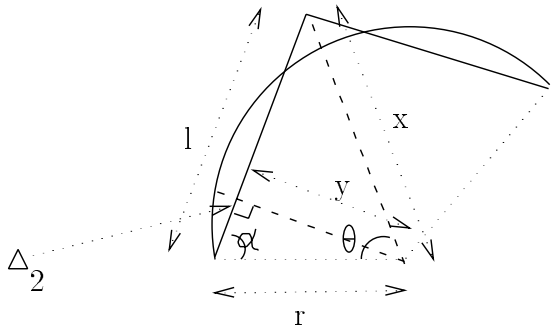


Figure 8.9: Δ_2 and y

$$\begin{aligned} &= r(\cos\theta + \sqrt{\cos\theta - (1 - \frac{l^2}{r^2})} \\ &= \frac{l}{\theta}(\cos\theta + \sqrt{\cos\theta - (1-\theta^2)}\ ) \\ &= \frac{l}{\theta}(\cos\theta + \sqrt{\theta^2 - \sin^2\theta}\ ) \end{aligned} \tag{8.3}$$

And we obtain

$$\Delta_1 = x - r = x - \frac{l}{\theta} = l \times \frac{\cos\theta + \sqrt{\theta^2 - \sin^2\theta} - 1}{\theta} \tag{8.4}$$

Δ_2 can also be expressed in terms of l and θ. Let y be the length of a perpendicular line from the arc center to the link (see Fig. 8.9). Then

$$\Delta_2 = r - y$$

and

$$y = r\sin\alpha \tag{8.5}$$

From Sine theorem

$$\frac{x}{\sin\alpha} = \frac{l}{\sin\theta}$$

Hence

$$\sin\alpha = \frac{x\sin\theta}{l} \tag{8.6}$$

By substituting (8.6) to (8.5)

$$y = r\frac{x\sin\theta}{l} = \frac{x\sin\theta}{\theta}$$
$$\Delta_2 = r - y = \frac{l}{\theta} - \frac{x\sin\theta}{\theta}$$

By substituting x in the above with (8.3), we get

$$\begin{aligned}\Delta_2 &= \frac{l}{\theta} - \frac{l}{\theta}(\cos\theta + \sqrt{\theta^2 - \sin^2\theta})\frac{\sin\theta}{\theta} \\ &= l \times \frac{1 - (\cos\theta + \sqrt{\theta^2 - \sin^2\theta})\frac{\sin\theta}{\theta}}{\theta}\end{aligned} \tag{8.7}$$

Now, let us evaluate the mapping errors Δ_1 and Δ_2 in (8.4) and (8.7) when r, the radius of arc to be approximated, changes (see Fig. 8.10). We consider the relative errors δ_1 and δ_2:

$$\begin{aligned}\delta_1 &= \frac{\Delta_1}{l} \\ \delta_2 &= \frac{\Delta_2}{l}\end{aligned}$$

Note that the relative errors δ_1 and δ_2 are functions of θ alone, as seen from (8.4) and (8.7). Fig. 8.11 is the graph for δ_1 and δ_2. In the graph, δ_1 and δ_2 are plotted as a function of $\frac{1}{\theta} = \frac{r}{l}$ instead of θ. It is to accentuate the fact that as the radius increases, the relative errors decrease, as expected from Fig. 8.10. From (8.2), the domain of the functions is $(\frac{1}{\pi}, \infty)$. As seen in the graph, the relative errors are below 20% when $r \geq 0.5 * l$.

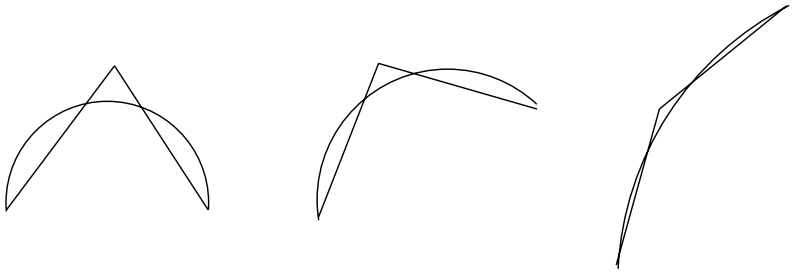


Figure 8.10: Increase r while Keeping l Constant

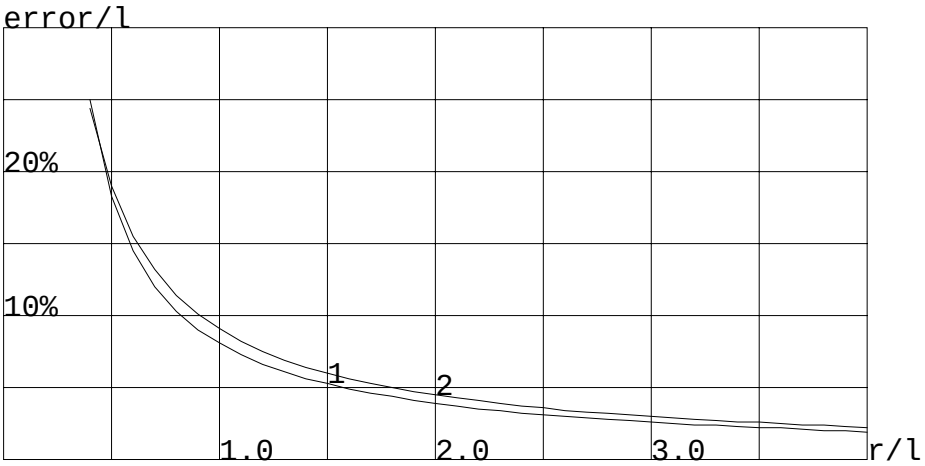


Figure 8.11: δ_1 and δ_2 as Function of $\frac{r}{l}$

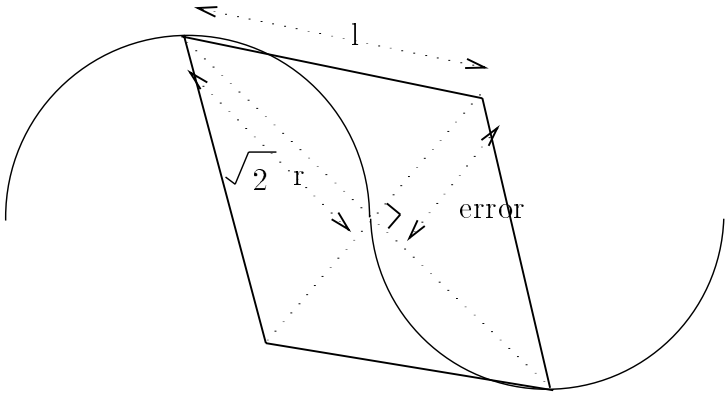


Figure 8.12: Mapping for Tangent Circles when $l = \frac{\pi}{2}r$

8.2.2 Tangent Arcs Case

For the tangent arcs case, an upper bound on the mapping errors which is obtained by replacing cubic spirals with tangent circular arcs is too conservative. It becomes conservative, because tangent circles can have different curvature, while cubic spirals have curvature continuity at tangent points. For example, the relative mapping error at the middle joint in Fig. 8.12 is:

$$\frac{error}{l} = \frac{\sqrt{(\frac{\pi}{2}r)^2 - (\sqrt{2}r)^2}}{\frac{\pi}{2}r} = 0.44$$

In this case, tangent circles have the same curvature with opposite signs ($\pm\frac{1}{r}$).

For this reason, we try to evaluate errors directly for tangent cubic spiral curves discussed in Section 6.4. Cubic spiral curves are specified by the size d and the deflection α (see Fig. 8.13). Hence, mapping errors should be evaluated for each pair of different cubic spirals (different d and α) which are tangent to each other and have zero curvature at the tangent point.

It is not necessary to enumerate all cases by stepping through the ranges of d, α while moving links along the curves generated. The critical cases

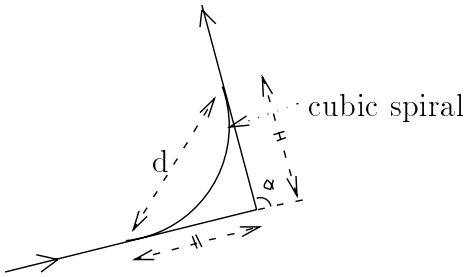


Figure 8.13: Size d and Deflection α of a Cubic Spiral. The same figure appears as Fig. 6.4

are limited to the cases which are somewhat similar to the one in Fig. 8.12 for the following reasons:

- If the size d increases without changing the deflection α, the error should decrease.
- If the end joints are not located at the apexes of cubic spirals, the case becomes more or less like a single arc case, whose error bound is, as we will see, more favorable.

The critical cases occur when the two cubic spirals have the same length:

$$l^{spiral} = 2l \tag{8.8}$$

and the two end joints of the two consecutive links are located at the apexes of cubic spiral curves. As easily imagined, the error for these cases becomes a monotonic function of α. The error increases as the deflection α increases and the turn becomes sharper.

In the above analysis, the maximum curvature constraint was ignored. It can be taken into account as follows. Given α and the maximum curvature κ_{max}, the lower bound for size d, d_{min}, of the cubic spiral is determined using

(6.4):

$$d \geq d_{min} = \frac{1.5\alpha D(\alpha)}{\kappa_{max}}$$

d_{min} thus obtained is not necessarily consistent with (8.8). For large α, the cubic spiral with size d_{min} may be longer than $2l$. When this is the case, we locate the end joints at distance l from the tangent point along the curve, and evaluate the error of mapping at the tangent point (Fig. 8.14). The error corresponding to these cases decreases as the deflection α increases. The is because size d satisfying the constraint increases as α increases.

Fig. 8.15 shows a graph for the errors obtained for the critical tangent arcs cases as a function of α, given the following three maximum curvature constraints.

$$\kappa_{max} = \begin{cases} \frac{1}{2.0l} \\ \frac{1}{1.0l} \\ \frac{1}{0.5l} \end{cases} \tag{8.9}$$

The relative error is plotted. Each error function decreases in a rage of α where the maximum curvature constraint becomes relevant to the error analysis. The maximum value for the error functions increases with κ_{max}, the maximum curvature constraint. As seen in the graph, the relative error does not exceed 22% for $\kappa_{max} = \frac{1}{1.0l}$.

8.2.3 Proposition for Error Bound

The results on the error bound are summarized as follows.

Proposition 3 *Let l be the length of each link of the jointed arm. The error of the every-other-joint mapping does not exceed $0.22*l$, if the following conditions are satisfied.*

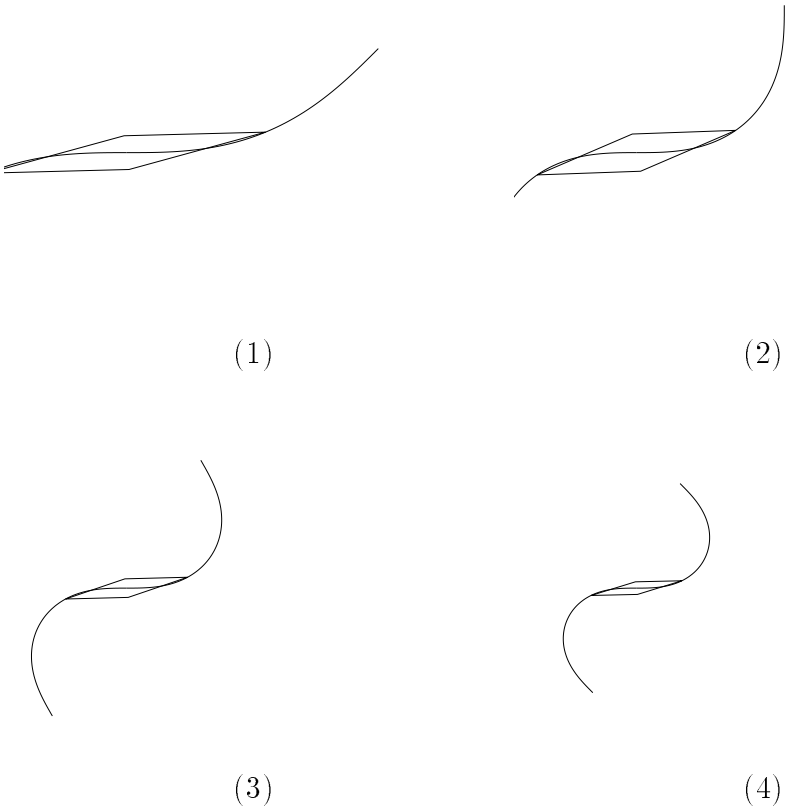


Figure 8.14: Tangent Arcs Case with Maximum Curvature Constraint. As deflection α increases, the link length l consistent with the maximum curvature constraint decreases, resulting in the mappings shown in the figure. Here the constraint is specified as: $\kappa_{max} = \frac{1}{1.0l}$.

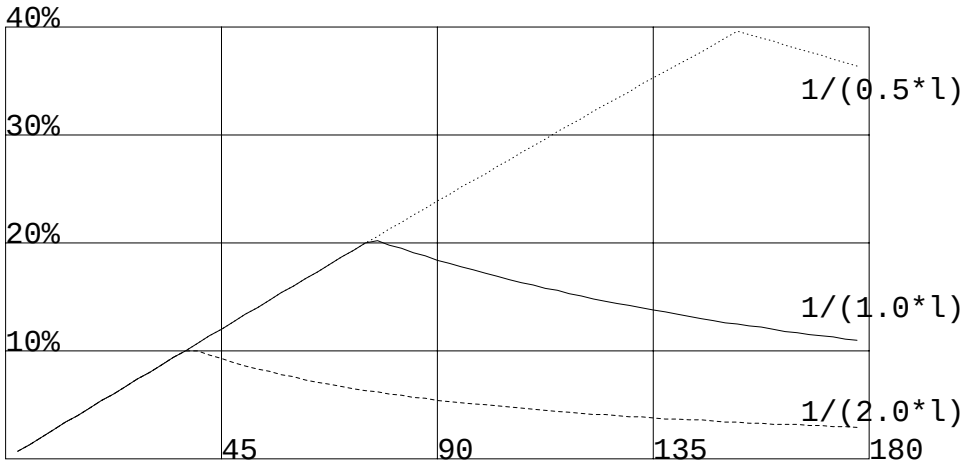


Figure 8.15: Relative Error for Tangent Arcs Case as Function of α

1. *Each cubic spiral segment (including the straight line segments at both ends, if they exist) is longer than* $2 * l$.

2. *The minimum turning radius of the path is* l*, or equivalently the maximum curvature of cubic spiral segments is* $1/l$.

Proof (sketch). The 22% error bound of the proposition comes from the tangent arcs case (Fig. 8.15). The first condition in the proposition is for restricting the cases to two cases: the single arc case and the tangent arcs case. The second condition comes from the tangent arcs case with the maximum curvature constraint: $\kappa_{max} = \frac{1}{1.0l}$ in (8.9). Under the second condition, the error for the single arc case is well below the 22% bound (see Fig. 8.11). □

Now the conditions for path planning have become explicit. In fact, the path shown in Fig. 7.1 was obtained by first growing the obstacles in Fig. 8.2 by $0.22 * l$ and then planning the path for the continuous manipulator with the above two conditions. The proposition guarantees that mapping the path for

the continuous manipulator back to a path for the jointed arm will yield a collision free path.

8.3 Improving the Approximation

Although the every-other-joint mapping may seem ad hoc, we believe it is a reasonable mapping scheme because of its efficiency and its easiness to obtain an error bound. Moreover, the mapping can easily be extended to every-third-joint mapping, or every-fourth-joint mapping.

In the every-third-joint mapping, we still have one degree of freedom left after we constrain the every third joints. The extra degree of freedom can be used to reduce the approximation error using an iterative minimization. Alternatively, it can be used to constrain the orientation of one of the three links.

But obtaining an explicit error bound for these mappings is difficult. Standard techniques for fitting or approximating data using a model with adjustable parameters (such as the least mean square fitting) have an objective similar to ours: approximating the continuous solution using a discrete model (a jointed arm) with adjustable variables (joint variables).

But these techniques are not directly applicable to our mapping problem because of the following reasons.

- We are interested in the maximal error, while standard curve fitting techniques focus on the average error or the sum of errors.
- Coordinates of points on a jointed arm are necessary to compute errors. But these coordinates are complex nonlinear functions (trigonometric functions) of the joint angles (θ_1,θ_2,...), known as forward kinematics.

- The minimization of the maximal error has to proceed iteratively, as is the case for a standard technique for fitting with nonlinear models [Press *et al.* 86]. This will be an expensive scheme.

What if we change our current task of reducing mapping errors to the original task: reaching the goal while avoiding obstacles ? We could use the artificial potential field [Khatib 86] to avoid obstacles. It may be possible to solve the local minima problem associated with the potential field approach by introducing appropriate subgoals (intermediate configurations) taken from the continuous solution. This seems to be an interesting problem to be investigated in the future.

8.4 Dynamic Simulation of the Swan's Neck Manipulator

Once a path has been obtained for a jointed arm, a dynamic simulation can be carried out using methods in the literature. Manipulators containing many DOFs do not cause a serious problem here.

In [Park 90], a dynamic simulation of a conceptual highly articulated manipulator has been performed, given the result of a mapping of the continuous trajectory to the articulated manipulator. The manipulator has 12 ball joints and 36 DOFs. Each link is defined as a cylinder. In computing the inertia tensor, it was assumed that the distribution of mass over the neck is uniform. The mass of a ball joint was neglected.

The dynamic simulation was done using the Newton-Euler formulation. One advantage of the Newton-Euler formulation is that its computational complexity is linear in terms of the degree of freedom of the manipulator [Craig 86, Luh *et al.* 80]. Trajectory planning was done in joint space. First

the joint angle velocities are determined. The end-point velocities in work space may then be calculated using the Jacobian matrix J.

Chapter 9

Summary and Conclusions

In the last chapter, we summarize our approach: path planning using the continuous model. Advantages and disadvantages of our approach will become clearer by comparing it with other approaches to path planning. We conclude the thesis by discussing the future work necessary to put highly redundant manipulators in work.

9.1 Summary

We have presented a path planning method for highly redundant manipulators by means of a continuous model, which captures a macroscopic shape of highly redundant manipulators.

The path planning problem has been shown to be $PSPACE$-complete in terms of DOF of the manipulator. Our approach overcomes the complexity with a strong heuristic: utilizing redundancy by means of the continuous model. The continuous model allows us to change the complexity of the planning problem from a function of both the DOF of the manipulator (believed to be exponential) and the complexity of the environment (polynomial), to a polynomial function of the complexity of the environment only.

The power of the continuous model comes from the ability to decompose the manipulator into segments, with the number, size, and boundaries of the segments varying smoothly and dynamically. First, we develop motion

schemas for the individual segments to achieve a basic set of goals in open and cluttered space. Second, we plan a smooth trajectory through free space for a *point robot* with a maximum curvature constraint, by searching a connectivity graph of primary convex regions. Third, the path generates a set of position subgoals for the continuous manipulator which are achieved by the basic motion schemas. Fourth, the mapping from the continuous model to the available jointed arm provides the curvature bound and obstacle envelopes required (in step 2) to guarantee a collision-free path.

DOF of the manipulator is a resource to be utilized in our approach, because the error bound on the mapping improves with the number of DOF of the manipulator. Essentially, we have transformed the problem of planning paths for highly redundant manipulators to the problem of finding smooth paths for point robots. The smooth path planning problem is a new subject in the field (see [Jacobs and Canny 89] for a related problem), and we expect improvements on the algorithm by using more computational geometry.

The validity for the continuous manipulator model is also supported by an extensive simulation we performed. While the simulation has been performed in 2-D, we have shown a natural extension to 3-D for each technique we have implemented for the 2-D simulation.

9.2 Comparison with Other Approaches

Advantages and disadvantages of our approach will become clearer by comparing it with other approaches to path planning.

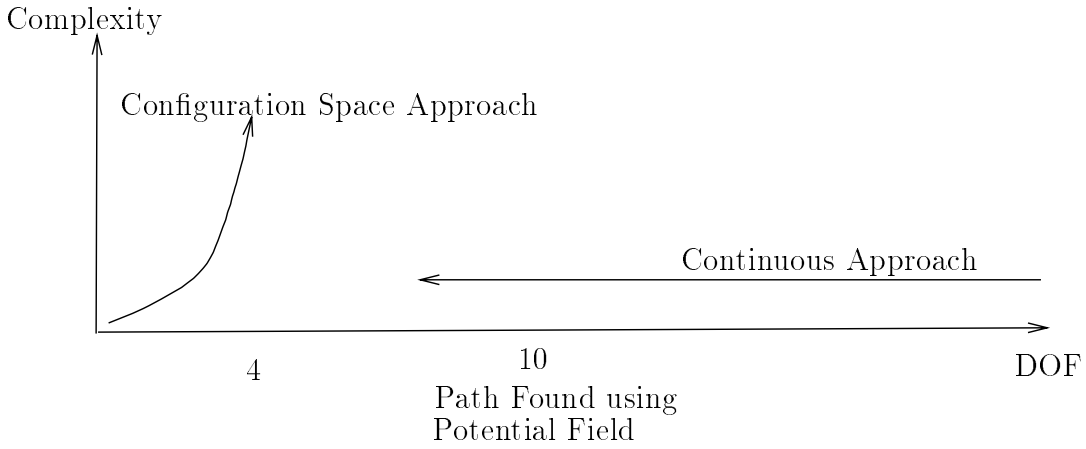


Figure 9.1: Three Approaches to Path Planning in terms of DOF

9.2.1 Complexity in terms of DOF

First, we compare the range of manipulator DOFs for which a collision free path can be found in a reasonable amount of time. Fig. 9.1 shows the comparison.

It is said in [Faverjon and Tournassoud 89] that the algorithms based on the configuration space approach can not be applied to manipulators with more than four degrees of freedom, because their complexity glows exponentially in terms of DOFs. We have seen in Section 2.3 that there are not any powerful and robust heuristics developed to utilize redundancy and overcome the exponential complexity.

It is difficult to evaluate the range of applicability of the artificial potential field approach because of its local minima problem. Hence, we took examples in [Barraquand and Latombe 89] which show the paths found for 8 DOF and 10 DOF manipulators. Those examples seem to show the leading edge of algorithms using the artificial potential field approach.

On the other hand, our approach to find a continuous solution first and then to approximate it by a jointed arm has a completely different range of

application, because its complexity does not depend on DOF of manipulators. As a matter of fact, as we increase the DOF of a manipulator, it becomes easier to find a path in terms of computation time. It is because two conditions for a continuous path, one on the maximum curvature and the other on the minimum segment length, are relaxed. We have shown paths found for a 12 DOF manipulator.

9.2.2 When We Fix DOF

Another interesting comparison is made by fixing the DOF of a manipulator and considering the complexity of path planning in terms of the complexity of its environment. It has been known the motion planning problem is tractable if we fix the DOF of a robot.

Let n be the complexity of an environment. Our algorithm runs in $O(n^{24})$ time. This time bound has been obtained using a loose upper bound on the number of primary convex regions and the worst case time complexity of A^* search. In practice, our algorithm is efficient, because the number of primary convex regions does not grow so rapidly [Rueb and Wong 87], and we have a strong heuristic for the search.

The theoretical exact algorithm in [Canny 88a], runs in $O(n^r \log n)$ time for a r DOF manipulator ($O(n^6 \log n)$ for a 6 DOF manipulator). Although the $o(n^6 \log n)$ complexity seems favorable, the algorithm has a large constant associated which is exponential in terms of DOF. This is why four is practically the upper limit of DOFs to be handled by the configuration space approach.

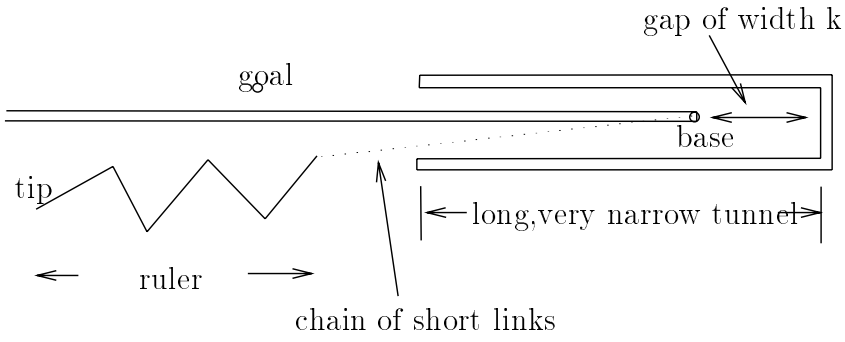


Figure 9.2: A Variation of the Ruler Folding Problem

9.2.3 Search Space for Our Approach

We are obviously using heuristics, although our algorithm is robust enough to find paths for highly redundant manipulators in reasonable environments. First, we assume there is enough open space around the base so that we can feed and retract the manipulator without difficulties. In addition, we are restricting the search space. The motion schema which we use primarily for obstacle avoidance is that of following trajectories in the task space (2-D or 3-D). As we have seen in Chapter 8, this motion schema corresponds to a limited class of trajectories in the joint space, those trajectories where subsequent joints follow the tip joint with some time delay. Furthermore, the joint rotations are limited by the maximum curvature constraint on the path searched for.

9.2.4 Ruler Folding Problem

Our algorithm fails when there is not a path to satisfy above restrictions. Consider the following problem taken from [Hopcroft *et al.* 85].

The jointed arm consists of a ruler and a chain of short links (Fig. 9.2). The chain links, shown in dotted lines in the figure, are short enough to turn freely inside the tunnel, while ruler links can turn very little inside the tunnel

because the tunnel is too narrow. In order for the tip of the ruler to reach its goal across an obstacle, we must fold the ruler within the length of k outside the tunnel to path through the gap in the tunnel.

It is easy to see that a path exists if and only if the ruler can be folded within the length of k. If the links of the ruler have integer lengths, the ruler folding problem is shown to be an NP-complete problem by a reduction from Partition Problem, another NP-complete problem [Garey and Johnson 79].

This problem may look too artificial. But we cannot find a path using the continuous approach, even if the length of the links in the ruler are equal and less than the gap width k. In this case, folding the ruler within the length k is trivially possible. However, in order to find a path using the continuous approach, we need more open space (more width and height) at the gap. We know from the result of our every-other-joint mapping that we need to turn using an arc segment whose center is at the base and whose length is at least twice of the link length of the ruler.

9.2.5 Advantage of Our Approach

Let us discuss more about the advantages of our approach compared to the configuration space approach.

- A representative algorithm [Lozano-Pérez 87a] of the configuration space approach is complete. It searches the joint space fully eventually, and finds a path if one exists. In this sense, the algorithm always succeeds.

  However the answer from the algorithm may be "No collision free path", and the task to reach its goal may fail. It is easy to imagine those cases when we get no-path answers for small DOF non-redundant manipulators

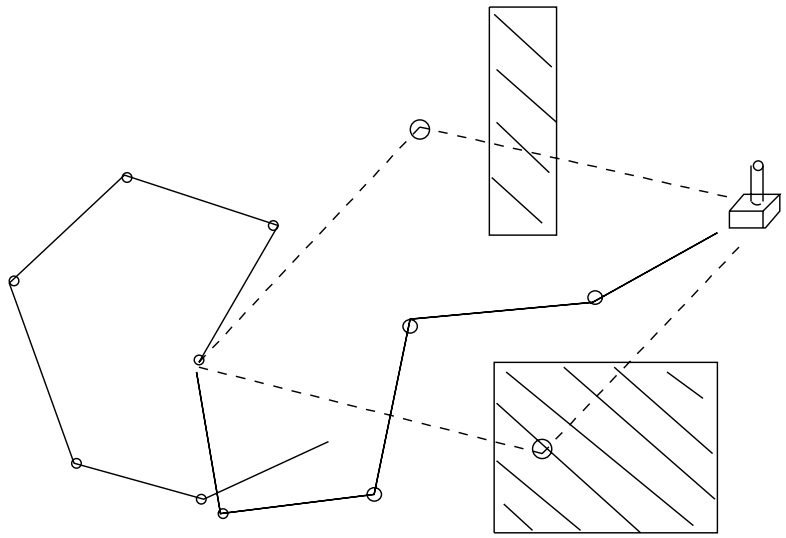


Figure 9.3: Redundancy Helps

which is the main concern of the configuration space methods. For a non-redundant manipulator, its configuration is totally determined by its end effector position/orientation, and this is why dexterious redundant manipulators are getting more attention to achieve tasks in hard-to-reach environments (Fig. 9.3).

- DOF of a manipulator acts as a benefit, not a cost in our approach. This is because the error bound on the mapping improves with the number of DOF of the manipulator. So the amount that obstacles are grown is decreased with increasing DOF.

- In practice, average-case complexity may be more important than worst-case complexity. For configuration space algorithms, the dominant factor in complexity is *constructing* the C-space map, the first step in the algorithms. So it is a up-front cost. They can still improve the average-case complexity by recursively increasing the resolution of the C-space map. However, for a given resolution, the cost for constructing the C-space

map is exponential in terms of DOF.

The average case complexity of our algorithm to find smooth paths using the heuristic search is much better than the worst case complexity because of the heuristic associated.

- Essentially, we have transformed the problem of planning paths for highly redundant manipulators to the problem of finding smooth paths for point robots. The smooth path planning problem is a new subject in the field and we expect improvements on the algorithm by using more computational geometry.

9.3 Future Work

9.3.1 3-D Simulation

We plan to extend our current simulator from 2-D to 3-D. We have already explained that a natural extension exists. There are two basic issues: the kinematics of the continuous manipulator model, and path planning.

The continuous manipulator model in 3-D is based on the differential geometry of space curves (Section 3.4), and is controlled by its curvature and its torsion. We use the Frenet equations to obtain the shape of the continuous model from curvature and torsion. Our hill climbing routine can be extended to solve open space problems in 3-D by changing curvature and torsion in a discrete way and by sampling typical configurations as we did with our 2-D simulator (see Section 4.5).

Two approaches are feasible for path-planning in 3-D space. Brooks [Brooks 83b] used a 2 1/2-D approach for pick-and-place operations, in which 3-D space is treated as a stack of 2-D spaces (see Section 7.2.1). Most of

the path planning routines implemented for the 2-D simulator can be used for the 2 1/2-D approach. Alternatively, we can decompose 3-D free space into primary convex regions. In Section 7.3.2), we have extended the decomposition algorithm presented in [Singh and Wagh 87] to 3-D.

While we have used cubic spiral curves to turn from one polygon to another in our 2-D simulator, we can use continuous 3-D curves with torsion to move from one polyhedron to another. The every-other-joint mapping can be applied to the continuous manipulator model in 3-D without any modification, although we need to re-establish the error bounding proposition in terms of curvature and torsion.

9.3.2 Building/Controlling a Highly Redundant Manipulator

So far, our research efforts have been devoted to studying geometrical aspects of highly redundant manipulators: kinematics and path planning. Building and controlling a highly redundant manipulator poses anther major challenge.

In literature on highly redundant manipulators, mechanical design of such manipulators is the main subject and some manipulators were actually built (Section 1.2). While designs using other exotic actuators are also proposed, tendon driven actuator systems seem to be popular. The tensor actuated elastic manipulator design in [Hirose *et al.* 83] is such a manipulator. In their paper, the design for the manipulator is made explicit with details.

We plan to build a highly redundant manipulator, not only because it is an interesting subject in its own right, but also the manipulator will provide us a test bed for controlling such devices.

In this thesis, only geometrical aspects of the motion planning have

been addressed. Moreover, we assumed complete information about the environment. It may be obtained from a CAD database of the mechanism in which a highly redundant manipulator works. The situation changes if complete information is not available.

We need to extend the current open-loop planning algorithm to a closed-loop method for planning and control based on sensory feedback from the environment. Although the use of the hill climbing search in path planning has been limited in this thesis, it will become mandatory when we try the feedback control in the incomplete information setting. While 3-D machine vision in general is still in a phase of an ambitious research program [Faugeras 89], Tokuta and Hughes' scanline algorithm to find primary convex regions from pixels [Tokuta and Hughes 90] can be used in 2-D experiments.

The problem of dynamics is central to controlling manipulators, and we have obtained some good results for a conceptual highly articulated manipulator (Section 8.4). At this moment, however, we do not have a clear idea what a control algorithm is like for highly redundant manipulators. Among the interesting works in control are the artificial potential field approach [Khatib 86] and declarative rational controllers [Kohn 90], and we need further investigations.

9.4 Contributions

Like some other research in robotics, notably Kuipers and Byun's research on robot spatial learning [Kuipers and Byun 88, Byun 90], our contributions are two fold: those which are practical and engineering oriented and those which are cognitive oriented.

While the concept of snake-like highly redundant manipulators to work in hard-to-reach places is not new, kinematics and path planning for such

manipulators have not been extensively studied so far. Our approach based on the continuous manipulator model provides a tractable path planning algorithm for such complex devices. Our approach also provides a possibility of speeding up the path planning process for manipulators, which is currently carried out off-line. Conceptually, path planning for highly redundant manipulators could be carried out as fast as path planning for point robots.

The question about the efficient path planning capability by humans for their arms has partially motivated our research (Section 1.1). We suspect that humans plan paths in their task space (2-D and 3-D) utilizing the redundancy of their body-arm system. Our result, although it is obtained for highly redundant manipulators, indicates that path planning, if carried out in this fashion, can be made very efficient.

Today's robot manipulators work only in well controlled and engineered environments. While there is a research robot to collect coke cans in offices [Connel 89], housework robots are still a dream. Although we do not notice in every day life, most housework involves geometrical problem solving. They are difficult and time consuming, if we try to apply conventional approaches to the problem. A seemingly simple job such as re-shelving books would be a formidable task, because its spatial map changes as the job is carried out. We believe that the our approach, using the continuous manipulator model for highly redundant manipulators, has a significant potential and is a promising solution to this sort of problem.

BIBLIOGRAPHY

[Ackley 89] D. H. Ackley. Associative learning via inhibitory search. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems.* Morgan Kaufmann Publishers, 1989.

[Andeen 88] G. B. Andeen, editor. *Robot Design Handbook.* McGraw-Hill Book Company, 1988.

[Arden 80] B. W. Arden, editor. *What can be automated? Computer Science and Engineering Research Study.* MIT Press, 1980.

[Arkin 89] R. C. Arkin. Navigational path planning for a vision based mobile robot. *Robotica*, 7, 1989.

[Asano 85] T. Asano, L. Guibas, J. Hershberger, and H. Imai. Visibility polygon search and euclidean shortest paths. In *Proceedings of 26th Symposium on foundations of Computer Science*, 1985.

[Barhen *et al.* 89] J. Barhen, S. Gulati, and M. Zak. Neural learning of constrained nonlinear transformations. *IEEE Computer*, pages 67–76, June 1989.

[Barnett 38] J. Barnett. *Transition Curves for Highways.* United States Department of Agriculture, Bureau of Public Roads, 1938.

[Barraquand and Latombe 89] J. Barraquand, and J.-C. Latombe. Robot motion planning: a distributed representation approach. Technical Report STAN-CS-89-1257, Department of Computer Science, Stanford University, May 1989.

[Barrow and Tenenbaum 78] H. Barrow and J. Tenenbaum. Recovering intrinsic scene characteristics from images. In A. Hanson and E. Riseman, editors, *Computer Vision Systems*. Academic Press, 1978.

[Basu and Aloimonos 90] A. Basu and J. Aloimonos. Approximate constrained motion planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1990.

[Brooks 83a] R. A. Brooks. Solving the find-path problem by good representation of free space. *IEEE transaction on Systems, Man and Cybernetics*, 13:190–197, 1983.

[Brooks 83b] R. A. Brooks. Planning collision-free motions for pick-and-place operations. *The International Journal of Robotics Research*, 2(4), 1983.

[Byun 90] Y.-T. Byun. *Spatial Learning Mobile Robots with a Spatial Semantic Hierarchical Model.* PhD thesis, Department of Computer Sciences, The University of Texas at Austin, 1990. also available as AI90-121, Artificial Intelligence Laboratory, The University of Texas at Austin.

[Canny 88a] J. F. Canny. *The Complexity of Robot Motion Planning.* The MIT Press, 1988.

[Canny 88b] J. Canny. Some algebraic and geometric computations in pspace. In *Proceedings of the ACM symposium on Theory of Computing*, 1988.

[Canny and Lin 90] J. F. Canny and M. C. Lin. An opportunistic global path planner. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1990.

[Chazelle 84] B. Chazelle. Convex partitions of polyhedra: A lower bound and worst-case optimal algorithm. *SIAM Journal of Computing*, 13(3):488–507, August 1984.

[Chirikjian and Burdick 90] G. S. Chirikjian and J. W. Burdick. An obstacle avoidance algorithm for hyper-redundant manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1990.

[Clement and Iñigo 90] W. I. Clement and R. M. Iñigo. Design of a snake-like manipulator. *Robotics and Autonomous Systems*, 6:265–282, 1990.

[Connel 89] J. H. Connel. *A Colony Architecture for an Artificial Creature.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1989. also available as AI TR-1151, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

[Craig 86] J. J. Craig. *Introduction to Robotics: Mechanics and Control.* Addison-Wesley, 1986.

[Drozda 84] T. J. Drozda. The spine robot... the verdict's yet to come. *Manufacturing Engineering*, pages 110–112, September 1984.

[Dubins 57] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[Dupont and Derby 86] P. E. Dupont and S. Derby. Planning collision free paths for redundant robots using a selective search of configuration space, 1986.

[Farin 90] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, Inc., 1990.

[Faugeras 89] O. D. Faugeras. A few steps toward artificial 3-d vision. In M. Brady, editor, *Robotics Science*. The MIT Press, 1989.

[Faverjon and Tournassoud 89] B. Faverjon and P. Tournassoud. A practical approach to motion-planning for manipulators with many degrees of freedom. In *The proceedings of the Fifth International Symposium on Robotics Research*, 1989.

[Fortune and Wilfong 88] S. Fortune and G. Wilfong. Planning constraint motion. In *Proceedings of the twentieth ACM symposium on theory of computing*, May 1988.

[Garey and Johnson 79] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W.H. Freeman and Company, 1979.

[Goldenberg *et al.* 85] A. A. Goldenberg, B. Benhabib, and R. G. Fenton. A complete generalized solution to the inverse kinematics of robots. *IEEE Journal of Robotics and Automation*, RA-1(1):14–20, March 1985.

[Gupta 90] K. K. Gupta. Fast collision avoidance for manipulator arms: A sequential search strategy. *IEEE Transactions on Robotics and Automation*, 6(5):522–532, October 1990.

[Hasegawa and Terasaki 88] T. Hasegawa and H. Terasaki. Collision avoidance: Divide-and-conquer approach by space characterization and intermediate goals. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(3):337–347, May/June 1988.

[Hemami 85] A. Hemami. Studies on a light weight and flexible robot manipulator. *Robotics*, 1:27–36, 1985.

[Hemami 90] A. Hemami. Joint velocity uniformity in redundant manipulators. *Robotica*, 8:69–72, 1990.

[Hirose *et al.* 83] S. Hirose, T. Kado, and Y. Umetani. Tensor actuated elastic manipulator. In *Proceedings of the Sixth World Congress on Theory of Machines and mechanisms*, 1983.

[Hirose and Morishima 90] S. Hirose and A. Morishima. Design and control of a mobile robot with an articulated body. *The International Journal of Robotics Research*, 9(2):99–114, April 1990.

[Hopcroft *et al.* 85] J. Hopcroft, D. Joseph, and S. Whitesides. On the movement of robot arms in 2-dimensional bounded regions. *SIAM Journal on Computing*, 14(2), May 1985.

[Horn 75] B. K. P. Horn. The fundamental eel equations. Technical Report Working paper No. 116, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, December 1975.

[Horn 83] B. K. P. Horn. The curve of least energy. *ACM Transactions on Mathematical Software*, 9(4):441–460, December 1983.

[Ikuta 90] K. Ikuta. Micro/miniature shape memory alloy actuator. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1990.

[Ikuta *et al.* 88] K. Ikuta, M. Tsukamoto, and S. Hirose. Shape memory alloy servo actuator system with electric resistance feedback and application

for active endoscope. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1988.

[Ivanescu and Badea 84] M. Ivanescu and I. Badea. Dynamic control for a tentacle manipulator. In *Proceedings of the International Conference on Robotics and Factories of the Future*, 1984.

[Jacak 89] W. Jacak. A discrete kinematic model of robots in the cartesian space. *IEEE Transactions on Robotics and Automation*, 5(4), August 1989.

[Jacobs and Canny 89] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

[Josin 88] G. Josin. Neural-space generalization of a topological transformation. *Biological Cybernetics*, 59:283–290, 1988.

[Jou and Han 90] E. D. Jou and W. Han. Minimal energy splines: I. plane curves with angle constraints. Technical Report UMIACS-TR-90-119, CS-TR-2533, Computer Science Center, university of Maryland, September 1990.

[Habib and Yuta 89] M. k. Habib and S. Yuta. Structuring free space as prime rectangular areas (pras) with on-line path planning and navigation for mobile robots. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, 1989.

[Kallay 87] M. Kallay. Method to approximate the space curve of least energy and prescribed length. *Computer-Aided Design*, 19(2):73–76, March 1987.

[Kanayama and Hartman 89] Y. Kanayama and B. I. Hartman. Smooth local path planning for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

[Khatib 86] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotic Research*, 5(1):90–98, 1986.

[Khosla and Volpe 88] P. Khosla and R. Volpe. Superquadratic artificial potentials for obstacle avoidance and approach. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1988.

[Kirćanski and Vukobratović 86] M. Kirćanski and M. Vukobratović. Contribution to control of redundant robotic manipulators in an environment with obstacles. *The International Journal of Robotics Research*, 5(4):112–119, 1986.

[Kohn 90] W. Kohn. Declarative multiplexed rational controllers. In *Proceedings of IEEE International Symposium on Intelligent Control*, 1990.

[Kokkinis and Wilson 88] T. Kokkinis and J. D. Wilson. Design of continuous robotic arms. In M. Jamshidi, J. Y. S. Luh, H. Seraji, and G. P. Starr, editors, *Robotics and Manufacturing*. ASME Press, 1988.

[Krogh and Thorpe 86] B. H. Krogh and C. E. Thorpe. Integrated path planning and dynamic steering control for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1986.

[Kuan *et al.* 85] D. T. Kuan, J. C. Zamiska, and R. A. Brooks. Natural decomposition of free space for path planning. In *Proceedings of IEEE*

International Conference on Robotics and Automation, 1985.

[Kuipers and Byun 88] B. Kuipers and Y.-T. Byun. A robust, qualitative method for robot spatial learning. In *Proceedings of the National Conference on Artificial Intelligence*, 1988.

[Lee and Lee 90] S. Lee and J. M. Lee. Multiple task point control of a redundant manipulator. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1990.

[Leven and Sharir 87] D. Leven and M. Sharir. An efficient and simple motion planning algorithm for a ladder moving in two-dimensional space amidst polygonal barriers. *Journal of Algorithms*, 8, 1987.

[Lowe 85] D. B. Lowe. Inspection and repair of nuclear plant. In *Handbook of industrial robotics*. Wiley, 1985.

[Lozano-Pérez 83b] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.

[Lozano-Pérez 87a] T. Lozano-Pérez. a simple motion-planning algorithm for general robot manipulators. *IEEE Journal of Robotics and Automation*, RA-3:224–238, June 1987.

[Lozano-Pérez 87b] T. Lozano-Pérez. Robot programming and artificial intelligence. In W. E. L. Grimson and R. S. Patil, editors, *AI in the 1980s and Beyond*. The MIT Press, 1987.

[Lozano-Pérez *et al.* 90] T. Lozano-Pérez, E. Mazer, J. L. Jones, and P. A. O'Donnell. Task-level planning of pick-and-place robot motions. In P. H. Winston and S. A. Shellard, editors, *Artificial Intelligence at MIT, Expanding frontiers*. The MIT Press, 1990.

[Luh *et al.* 80] J. Y. Luh, M. W. Walker, and R. P. Paul. On-line computational scheme for mechanical manipulators. *IEEE Transactions on Automatic Control*, 25(3), 1980.

[Maciejewski and Klein 85] A. A. Maciejewski and C. A. Klein. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, 4(3):109–117, 1985.

[Martelli 77] A. Martelli. On the complexity of admissible search algorithms. *Artificial Intelligence*, 8:1–13, 1977.

[Mel 90] B. W. Mel. *Connectionist Robot Motion Planning.* Academic Press, Inc., 1990.

[Morecki *et al.* 87] A. Morecki, K. Jaworek, W. Pogorzelski, T. Zielinska, J. Fraczek, and G. Malczyk. Robotics system - elephant trunk type elastic manipulator combined with a quadruped walking machine. In *Proceedings of the Second International Conference on Robotics and Factories of the Future*, 1987.

[Nelson 89] W. Nelson. Continuous-curvature paths for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

[Nilsson 80] N. J. Nilsson. *Principles of Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., 1980.

[Nutbourne *et al.* 72] A. W. Nutbourne, P. M. McLellan, and R. M. Kensit. Curvature profiles for plane curves. *Computer-Aided Design*, 4(4):176–184, 1972.

[O'Dunlaing and Yap 85] C. O'Dunlaing and C. Yap. A 'retraction' method for planning the motion of a disc. *Journal of Algorithms*, 6, 1985.

[Pal 78a] T. K. Pal. Intrinsic spline curve with local control. *Computer-Aided Design*, 10(1):19–29, January 1978.

[Pal 78b] T. K. Pal. Mean tangent rotational angles and curvature integration. *Computer-Aided Design*, 10(1):31–34, January 1978.

[Pal and Nutbourne 77] T. K. Pal and A. W. Nutbourne. Two-dimensional curve synthesis using linear curvature elements. *Computer-Aided Design*, 9(2):121–134, April 1977.

[Park 90] J. Park. Dynamic simulation of swan's neck. Unpublished Manuscript, March 1990.

[Pettinato and Stephanou 89] J. S. Pettinato and H. E. Stephanou. Manipulability and stability of a tentacle based robot manipulator. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, 1989.

[Pieper 68] D. L. Pieper. *The kinematics of manipulators under computer control.* PhD thesis, Stanford University, Mechanical Engineering Department, 1968.

[Press *et al.* 86] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes.* Cambridge University Press, 1986.

[Rao and Arkin 90] T. Rao and R. C. Arkin. 3d navigational path planning. *Robotica*, 8, 1990.

[Reif 79] J. H. Reif. Complexity of the generalized movers' problem. In *Proceedings of the 20th IEEE symposium on Foundations of Computer Science (San Juan, Puerto Rico)*, 1979.

[Rimon and Koditschek 89] E. Rimon and D. E. Koditschek. The construction of analytic diffeomorphisms for exact robot navigation on star worlds. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

[Ronse 89] C. Ronse. A bibliography on digital and computational convexity (1961-1988). *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2(2):181–190, February 1989.

[Rueb and Wong 87] K. D. Rueb and A. K. C. Wong. Structuring free space as a hypergraph for roving robot path planning and navigation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 9(2):263–273, 1987.

[Schechter 78a] A. Schechter. Synthesis of 2d curves by blending piecewise linear curvature profiles. *Computer-Aided Design*, 10(1):8–18, January 1978.

[Schechter 78b] A. Schechter. Linear blending of curvature profiles. *Computer-Aided Design*, 10(2):101–109, April 1978.

[Schwartz and Sharir 83] J. T. Schwartz and M. Sharir. On the piano movers' problem: II. general techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4, 1983.

[Schwartz and Sharir 88] J. T. Schwartz and M. Sharir. A survey of motion planning and related geometric algorithms. *Artificial Intelligence*,

37:157–169, 1988.

[Shahinpoor *et al.* 86] M. Shahinpoor, H. Kalhor, and M. Jamshidt. On magnetically activated robotic tensor arms. In *Proceedings of the International Symposium on Robot Manipulator: Modeling, Control, and Education*, 1986.

[Sharir and Schorr 84] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. In *Proceedings of the 16th annual ACM symposium on Theory of Computing*, 1984.

[Singh and Wagh 87] J. S. Singh and M. D. Wagh. Robot path planning using intersecting convex shapes: Analysis and simulation. *IEEE journal of Robotics and Automation*, RA-3(2):101–108, April 1987.

[Stoker 69] J. J. Stoker. *Differential Geometry.* Wiley-Interscience, 1969.

[Taylor *et al.* 82] R. H. Taylor, P. D. Summers, and J. M. Meyer. Aml: A manufacturing language. *International Journal of Robotics Research*, 1(3), Fall 1982.

[Taylor *et al.* 83] W. Taylor, D. Lavie, and I. Esat. A curvilinear snake arm robot with gripper-axis fiber-optic image processor feedback. *Robotica*, 1:33–39, 1983.

[Todd 86] D. J. Todd. *Fundamentals of robot technology.* John Wiley and Sons, 1986.

[Tokuta and Hughes 90] A. Tokuta and K. Hughes. Scanline algorithms in robot path planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1990.

[Waldron *et al.* 87] K. J. Waldron, V. Kumar, and A. Burkat. An actively coordinated mobility system for a planetary rover. In *Proceedings of 1987 international Conference on Advanced Robotics*, 1987.

[Warren 89] C. W. Warren. Global path planning using artificial potential field. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

[Welzl 85] E. Welzel. Constructing the visibility graph for n line segments in $o(n^2)$ time. *Information Processing Letters*, 20:167–172, 1985.

[Wilfong 88] G. T. Wilfong. Motion planning for an autonomous vehicle. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1988.

[Wilfong 89] G. Wilfong. Shortest paths for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

[Wilson 83] J. F. Wilson. Robotic mechanics and animal morphology. In M. Brady, L. A. Gerhardt, and H. F. Davidson, editors, *Robotics and Artificial Intelligence*, NATO ASI series. Springer-Verlag, 1983.

[Yap 87] C.-K. Yap. Algorithmic motion planning. In *Algorithmic and geometric aspects of robotics*, volume 1 of *Advances in robotics*, chapter 3. L. Erlbaum Associates, 1987.

[Yeung and Gekey 89] D.-Y. Yeung and G. A. Gekey. Using a context-sensitive learning network for robot arm control. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

[Yoshikawa 84] T. Yoshikawa. Analysis and control of robot manipulators with redundancy. In M. Brady and R. Paul, editors, *Robotics research: the first international symposium*, MIT Press series in artificial intelligence. MIT Press, 1984.

VITA

Akira Hayashi was born in Yamaguchi city, Japan, on November 16, 1950, the son of Etsuko and Shigesada Motonaga. After completing his work at Shinjuku High School, Tokyo, Japan, in 1969, he entered Kyoto University, Kyoto, Japan. He received the degree of Bachelor of Science in Mathematics from Kyoto University in March 1974. After graduation, he worked for IBM Japan as a systems engineer in Data Center Service department.

In August 1986, he entered Brown University and obtained the degree of the Master of Science in Computer Science in May 1988. In August 1988, to seek a career in research, he left IBM Japan and entered the doctoral program in the Department of Computer Sciences, the University of Texas at Austin.

Permanent address: 2117 Kamigo
Ogori-cho, Yoshiki-gun
Yamaguchi Prefecture 754
Japan

This dissertation was typeset[1] with LaTeX by the author.

[1] LaTeX document preparation system was developed by Leslie Lamport as a special version of Donald Knuth's TeX program for computer typesetting. TeX is a trademark of the American Mathematical Society. The LaTeX macro package for The University of Texas at Austin dissertation format was written by Khe-Sing The.