

**THEORY AND APPLICATIONS
OF
MODULAR RECONFIGURABLE ROBOTIC SYSTEMS**

Thesis by
I-MING CHEN

**In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy**

**California Institute of Technology
Pasadena, California**

1994

(Submitted March 14, 1994)

©1994
I-Ming Chen
All rights Reserved

Acknowledgements

First, I would like to express my deepest gratitude to my advisor, Joel Burdick, for his constant support and encouragement over the past five years. Joel's broad knowledge and enthusiasm for everything we have worked on have been an inspiration for me, and I hope I can follow his example as I start my own professional career. I have also learned from him to be patient and open-minded toward everything. These are the things that cannot be found in any textbook.

I am grateful to the members of my examining committee, Professors Richard Murray, Erik Antonsson, Fred Culick, and Dr. Guillermo Rodriguez, for taking their valuable time to familiarize themselves with my thesis, and for constructive comments. In addition, I would like to thank Dr. Rodriguez for motivating this work. Richard Murray's helpful advice and comments provided me more insight in this research work. I also enjoyed discussing and receiving comments from Dr. Elon Rimon, now a professor of Technion University in Israel, and Dr. Greg Chirikjian, my long time colleague and friend.

Many thanks to members of the robotics laboratory at Caltech. The computation assistance from Howie Choset and Jim Ostrowski were excellent. Endless conversations, questions, and answers from Brett Slatkin and Andrew Lewis will be memorable. I also want to thank my fellow graduate students, Chih-Yung Wen, Chung-Yu Mou, Wen-Jean Hsueh, Angela Shih, Eugene Lipovetsky, Petr Pich, and Jung-Chih Chiao, and the resourceful staffs, Cecilia Lin, Jackie Beard, Dana Young, and Charmaine Boyd of Thomas Lab. Their friendship and help maintained my sanity and vigor.

I am greatly in debt to all my family members. My dearest parents are the strongest supporters for my graduate student life emotionally and intellectually. Without their open minds and constant discipline, I could not have been what I am now. I also want to express my gratitude to my uncle and aunt, Joe and Lyn, for their constant care and love during my stay in Caltech. I owe a lot to my sisters, Marina and Margaret, for giving advice to their youngest brother in every aspect of my life ranging from

academic research to making a girlfriend. Finally, I would like to dedicate this thesis to my beloved grandmother, who passed away in March 1993. She watched me growing up since I was first born. It will always be a sorrow for me that she was unable to see her grandson finish his Ph.D.

I would also like to acknowledge the financial support that I received during my stay here. My research has been funded in part by Caltech's President's Fund and the National Science Foundation, under grant MSS-9157843.

Abstract

A modular reconfigurable robotic system consists of various link and joint units with standardized connecting interfaces that can be easily separated and reassembled into different configurations. Compared to a fixed configuration robot, which is usually a compromised design for a limited set of tasks, a modular robot can accomplish a large class of tasks through reconfiguration of a small inventory of modules. This thesis studies how to find an optimal module assembly configuration constructed from a given inventory of module components for a specific task. A set of generalized module models that bear features found in many real implementations is introduced. The modular robot assembly configuration is represented by a novel *Assembly Incidence Matrix* (AIM). Equivalence relations based on module geometry symmetries and graph isomorphisms are defined on the AIMs. An enumeration algorithm to generate non-isomorphic assembly configurations based on this equivalence relation is proposed. Examples demonstrate that this method is a significant improvement over a brute force enumeration process. Configuration independent kinematic models for modular robots are developed, and they are essential for solving the task-optimal configuration problem. A task-oriented objective function is defined on the set of non-isomorphic module assembly configurations. Task requirements and kinematic constraints on the robot assembly are treated as parameters to this objective function. The task-optimal configuration problem is formulated as a combinatorial optimization problem to which genetic algorithms are employed for solutions. Examples of finding task-optimal serial revolute-jointed robot configurations are demonstrated. In addition, the applications of modular robots to planning multifinger grasping and manipulation are developed. Planning two-finger grasps is done through finding antipodal point grasps on smooth shaped objects. Planning n-finger grasps is achieved by defining a qualitative force-closure test function on the n-finger grasps on an object. Applications of this test function to manipulation task and finger gaiting are illustrated.

Table of Contents

| | |
|--|-----|
| Acknowledgements | iii |
| Abstract | v |
| 1. Introduction and Motivation | 1 |
| 1.1. Past Design Efforts | 3 |
| 1.2. Module Assembly Issues | 6 |
| 1.3. Organization of This Work | 8 |
| 2. Conceptual Model of Modules | 12 |
| 2.1. Joint Modules | 13 |
| 2.2. Link Modules | 14 |
| 2.3. Module Dimensions | 17 |
| 2.4. Discussion | 18 |
| 3. Mathematics for Modular Robotics | 19 |
| 3.1. Equivalence, Groups, and Permutations | 20 |
| 3.2. Symmetric Rotations | 22 |
| 3.3. Pólya's Counting Theorem | 24 |
| 3.4. Graphs | 28 |
| 3.5. Kinematic Graphs | 32 |
| 3.6. Discussion | 35 |
| 4. Enumeration of Assembly Configurations | 36 |
| 4.1. Enumerating Joint Assemblies on Links | 37 |
| 4.1.1. Assembly Patterns | 38 |
| 4.1.2. Algorithms for Listing Distinct Assembly Patterns | 41 |
| 4.1.3. A Note on Incorporating R-Joints with Joint Limit | 43 |
| 4.2. Assembly Incidence Matrices | 45 |

| | | |
|-----------|--|------------|
| 4.2.1. | Equivalence of AIMS | 47 |
| 4.2.2. | Hashed Assembly Incidence Matrix | 52 |
| 4.3. | Enumerating Robot Assembly Configurations | 54 |
| 4.3.1. | The Enumeration Algorithm | 55 |
| 4.3.2. | Examples | 59 |
| 4.3.3. | Computational Complexity Issues | 63 |
| 4.4. | Closed-Loop Construction Enumeration | 67 |
| 4.5. | Discussion | 67 |
| 5. | Modular Robot Kinematics | 69 |
| 5.1. | Forward Kinematics | 71 |
| 5.1.1. | Single Link Kinematics | 73 |
| 5.1.2. | Dyad Kinematics | 76 |
| 5.1.3. | Tree Robot Forward Kinematics Algorithm | 82 |
| 5.1.4. | Forward Kinematics Examples | 88 |
| 5.2. | Kinematic Equivalence | 89 |
| 5.2.1. | Equivalence of R-joint Serial Modular Robots | 91 |
| 5.2.2. | Equivalence Test Procedure | 95 |
| 5.3. | Inverse Kinematics | 98 |
| 5.3.1. | Derivation of the End Link Jacobian | 102 |
| 5.4. | Discussion | 104 |
| 6. | Task-Oriented Optimal Configurations | 105 |
| 6.1. | General Framework | 106 |
| 6.2. | Task Specifications | 109 |
| 6.2.1. | Definition of Robot Tasks | 109 |
| 6.2.2. | Task Evaluation Criteria | 110 |
| 6.3. | Structure Specifications | 112 |
| 6.3.1. | DOF Selection | 112 |

| | | |
|-----------|--|------------|
| 6.3.2. | Topology Selection | 113 |
| 6.3.3. | Module Assembly Preference | 114 |
| 6.4. | Assembly Configuration Evaluation Function of Serial Robots..... | 118 |
| 6.4.1. | Workspace Check Procedure | 120 |
| 6.5. | Genetic Algorithms for Modular Robots..... | 121 |
| 6.5.1. | Coding Schemes for AIMS | 123 |
| 6.5.2. | GA for Task-Optimal Configuration Problem | 125 |
| 6.6. | Examples | 127 |
| 6.7. | Discussion..... | 131 |
| 7. | Planning Multifinger Hand Grasps | 132 |
| 7.1. | Contact Configuration Space | 134 |
| 7.2. | Force-Closure Grasps | 136 |
| 7.2.1. | Two-Finger Force-Closure Grasps on Planar Objects | 137 |
| 7.2.2. | N-Finger Force-Closure Grasps on Planar Objects | 139 |
| 7.2.3. | Force-Closure Grasps on Spatial Objects | 142 |
| 7.3. | Two-Finger Grasp Planning | 142 |
| 7.3.1. | Planar Objects | 143 |
| 7.3.1.1. | Force-Closure Regions in Contact C-Space | 143 |
| 7.3.1.2. | A Grasping Energy Function | 143 |
| 7.3.1.3. | Planning Antipodal Point Grasps | 145 |
| 7.3.1.4. | Representations of Planar Objects | 147 |
| 7.3.2. | Spatial Objects | 149 |
| 7.3.2.1. | Representation of Spatial Objects | 150 |
| 7.4. | N-Finger Grasp Planning | 154 |
| 7.4.1. | A Qualitative Force-Closure Test Function | 154 |
| 7.4.2. | Properties and Symmetries of the FC-Surfaces | 157 |
| 7.4.2.1. | Type I FC-Surfaces | 157 |

| | |
|---|------------|
| 7.4.2.2. Type II FC-Surfaces | 158 |
| 7.4.3. Force-Closure Contact Modes | 159 |
| 7.4.3.1. Definition | 159 |
| 7.4.3.2. Identifying FC-Contact Modes in a Grasp | 161 |
| 7.4.4. Characterization of the n -finger Force-Closure Sets | 162 |
| 7.4.5. Application to Complex Multifinger Manipulation | 166 |
| 7.4.5.1. Multifinger Manipulation | 166 |
| 7.4.5.2. Finger Gaits | 169 |
| 7.4.5.3. Dextrous Manipulation Example | 169 |
| 7.4.5.4. Dextrous Manipulation with Sliding | 170 |
| 7.4.6. Issues Regarding N-Finger Grasps on 3-D Objects | 172 |
| 7.5. Discussion | 173 |
| 8. Conclusions | 175 |
| References | 180 |

List of Figures

| | | |
|-------|--|----|
| 1.1. | Revolute and prismatic joint modules (after [17]) | 5 |
| 1.2. | Two configurations of UT's modular robots (after [17]) | 5 |
| 1.3. | Schematic diagram for module assembly problem | 9 |
| 2.1. | Types of joints | 14 |
| 2.2. | Link modules—a cubic box and a prism | 16 |
| 2.3. | Real implementation of UT's connectors | 16 |
| 2.4. | Module models of the connectors | 17 |
| 2.5. | Joint dimension | 18 |
| 2.6. | Link dimensions | 18 |
| 3.1. | Symmetric rotation about z-axis by 90° | 23 |
| 3.2. | A labeled graph and a specialized graph | 29 |
| 3.3. | Isomorphism and automorphism of graphs | 31 |
| 3.4. | The Watt's linkage and its kinematic graph | 33 |
| 3.5. | A homogeneous modular robot and its graph | 34 |
| 3.6. | A hybrid modular robot and its graph | 35 |
| 4.1. | Three assembly patterns on a prism | 37 |
| 4.2. | Distinct patterns for a prism with 1 R- and 1 H-joint | 41 |
| 4.3. | Forbidden sector of an R-joint with joint limit | 43 |
| 4.4. | Forbidden sectors in different orientations | 43 |
| 4.5. | Assembly pattern with an R-joint and an R_l -joint | 45 |
| 4.6. | Relative orientations of two prisms with a P-joint | 47 |
| 4.7. | Three physically identical configurations | 48 |
| 4.8. | The flow chart of RobotEnumerate | 57 |
| 4.9. | Two non-isomorphic specialized graphs | 60 |
| 4.10. | Distinct configurations of a 3-link 2-DOF hybrid robot | 61 |

| | |
|---|-----|
| <i>List of Figures</i> | xi |
| 4.11. The graph of a 3-DOF serial modular robot | 63 |
| 4.12. Distinct assembly configurations of the 3-DOF robot | 63 |
| 4.13. 3-DOF fixed base robot example (continued) | 64 |
| 4.14. 3-DOF fixed base robot example (continued) | 65 |
| 4.15. 3-DOF fixed base robot example (continued) | 66 |
| 4.16. Eight non-isomorphic closed-loop configurations | 68 |
| 5.1. A rigid body displacement g | 71 |
| 5.2. The connecting line associated with port i | 74 |
| 5.3. A schematic kinematic graph | 76 |
| 5.4. Part of the module assembly | 77 |
| 5.5. Situation for indeterminate R_{ij} | 80 |
| 5.6. Kinematic graph of a quadruped \tilde{G} | 89 |
| 5.7. Forward kinematics examples | 89 |
| 5.8. Kinematically equivalent robots | 90 |
| 5.9. Symmetric rotation on the end link | 92 |
| 5.10. Two kinematically equivalent planar robots | 94 |
| 5.11. Joint axes parameters | 95 |
| 5.12. 3-DOF kinematically equivalent robots (A) | 96 |
| 5.13. 3-DOF kinematically equivalent robots (B) | 97 |
| 5.14. Block diagram of the NIK (after [47]) | 99 |
| 5.15. Relations among frames w , w' , and e | 103 |
| 6.1. Framework for task-optimal configuration problem | 107 |
| 6.2. Definition of a Robot Task | 110 |
| 6.3. Assembly patterns with joint redundancy | 115 |
| 6.4. MAP for assembly patterns on a prism and a cube | 117 |
| 6.5. Structure of ACEF for serial modular robots | 119 |
| 6.6. An assembly string representation of an AIM | 124 |

| | | |
|-------|--|-----|
| 6.7. | GA for task-optimal configurations | 126 |
| 6.8. | Initial configurations (6.6) | 128 |
| 6.9. | Final configurations (6.6) | 128 |
| 6.10. | Average and maximum fitness in every generation (6.6) | 128 |
| 6.11. | Initial configurations (6.7) | 129 |
| 6.12. | Final configurations (6.7) | 129 |
| 6.13. | Average and maximum fitness in every generation (6.7) | 129 |
| 6.14. | Initial configurations (6.8) | 130 |
| 6.15. | Final configurations (6.8) | 130 |
| 6.16. | Average and maximum fitness in every generation (6.8) | 130 |
| 7.1. | Friction cone at contact u_i | 138 |
| 7.2. | Squeezing grasp | 139 |
| 7.3. | Expanding grasp | 139 |
| 7.4. | Supporting plane of $CO(W(\mathbf{q}))$ | 142 |
| 7.5. | A planar object | 148 |
| 7.6. | FC-region ($\mu = 0.3$) | 148 |
| 7.7. | Grasping energy function E | 149 |
| 7.8. | A spatial object parameterized by spherical product | 153 |
| 7.9. | From different viewpoint | 153 |
| 7.10. | FC-curves and FC-regions of an ellipse | 156 |
| 7.11. | FC-surfaces for disk example | 164 |
| 7.12. | Constant θ_3 slices of C_3 | 164 |
| 7.13. | $\theta_3 = 0$ slices of C_3 with variable μ | 166 |
| 7.14. | A planar three-finger system | 167 |
| 7.15. | FC-2 contact regions and the trajectories of the contact points on the ellipse . . | 171 |
| 7.17. | Finger 1 slides while Fingers 2 and 3 are force-closure | 171 |
| 7.16. | Snapshots of a dextrous manipulation motion sequence | 172 |
| 7.18. | Polygonal approximation of a 3-D friction cone | 173 |

List of Tables

| | | |
|------|---|-----|
| 3.1. | Symmetric rotation group \mathcal{R}_L of a prism L | 24 |
| 3.2. | Symmetric rotation group \mathcal{R}_B of a cube B | 25 |
| 4.1. | Assembly patterns in Fig. 4.1 | 38 |
| 4.2. | Assembly patterns for R_l -joints in Fig. 4.4 | 44 |
| 4.3. | Comparison between two algorithms..... | 62 |
| 5.1. | PCT for prism module (L) | 75 |
| 5.2. | PCT for cubic box module (B) | 75 |
| 5.3. | IPT for Cube — Cube | 82 |
| 5.4. | IPT for Prism — Prism | 83 |
| 5.5. | IPT for Cube — Prism | 84 |
| 5.6. | IPT for Prism — Cube | 85 |
| 6.1. | The manipulability of task points | 112 |
| 6.2. | Task point set (6.6) | 127 |
| 6.3. | Task point set (6.7) | 128 |
| 7.1. | Extrema and antipodal points on the planar object | 148 |
| 7.2. | Extrema and antipodal points on the spatial object | 152 |

Chapter 1

Introduction and Motivation

The notion of modularity in design has long existed in many engineering disciplines, such as VLSI design [63], computer circuit board design, and software design. The main philosophy behind these approaches is to divide a complicated system into different functional modules with high portability, ease of maintenance, and logical clarity.

From mechanical perspective, a robotic system is a collection of connected joints and links which is to be employed for a particular set of tasks. The performance of a conventional robotic mechanism, e.g., workspace, dexterity, and loading condition, etc., are determined by its kinematic parameters, such as link lengths, joint positions, and types of joints, and its structural topology. In industrial robot design, the designer chooses these factors during the initial design phase so as to satisfy the given requirements of a set of tasks. The design process often becomes a time consuming and expensive process, and the outcome is a compromised device for the task requirements. From the task point of view, the versatility of such system is only for a limited class of tasks and sometimes becomes an over-design for a subset of less complex tasks. For example, a 6-DOF Puma manipulator with a relatively long reach in all directions in the operation space is suitable for painting, welding, and parts handling. On the other hand, a horizontal 4-DOF SCARA type manipulator, connected with relatively short links, is suitable for delicate table-top assembly operations requiring accuracy and selective

stiffness. Using the Puma robot to perform an IC assembly task becomes a trade-off between versatility and cost-efficiency. In some situations, using robots with different parameters for a variety of task is possible when the task requirements are specified in advance. However, in many unstructured and less predictable environments, such as a nuclear waste retrieval site, aboard a space station, or a lunar base construction site, it is very difficult or impossible to design a single robotic system that can meet a wide range of task requirements.

In these circumstances, it might be advantageous to deploy a modular reconfigurable robotic system which can be reconfigured itself into robots with different parameters which are individually well suited to the diverse task requirements. By a modular reconfigurable robotic system, we mean one in which various sub-assemblies with standardized connecting interface design, at the level of joints and links, can be easily separated and reassembled into different configurations. Such a robotic system has several advantages over conventional fixed-parameter design. First of all, it is very flexible and adaptable to different tasks requirements and working environments. The standardized design is easy for maintenance, modification, and transportation, and economical for manufacturing. Furthermore, it allows the re-use of the same part for different purposes, and thus reduces the total inventory of modules and the cost of redesign and repair.

There are several challenging technological and theoretical research issues that need to be addressed before putting such modular reconfigurable robotic systems into work. In order to achieve configuration independence, a modular robotic system must extend the notion of modularity to include mechanical design, electronics design, control algorithms, software, and communications. The most fundamental issue is the hardware design, which includes module design, mechanical and electronic interface design. Standard module component design is necessary for the maintenance of the integrated system. Standard interface design allows modules to be interchanged without incompatibility. Researchers at University of Stuttgart, Carnegie Mellon University, University of Toronto, Nagoya University, and University of Texas [17,32,87,91,103] have built

modular robotic systems to address this issue. Configuration independent software and control architectures [34,90] and communication methods [94] are also very important in controlling the system to carry out a task. Another important area is the module assembly planning problem which is the main theme of this thesis. As will be shown in the following chapters, with a set of standard components in hand, a methodology to enumerate all possible arrangements of the modules and to find an optimal module assembly for a specific task is definitely necessary. The methodology developed here is a completely general approach and can be applied to all existing modular robot systems.

The remainder of this chapter is organized as follows. Section 1.1 reviews previous efforts in the design and implementation of modular reconfigurable robotic systems. Section 1.2 discusses the module assembly issues and problem solving framework. Section 1.3 outlines the organization of the remaining chapters in this thesis.

1.1. Past Design Efforts

Several prototype modular robotic systems have actually been built and demonstrated, including the “Reconfigurable Modular Manipulator System” (RMMS) developed by Khosla and co-workers at CMU [87], the several generations of the “Cellular robotic system” (CEBOT) developed by Fukuda and co-workers at Nagoya University [32], and modular manipulator systems developed by Cohen et al. at University of Toronto [17], by Wurst at University of Stuttgart [103], and by Tesar and Butler in University of Texas at Austin [91].

The RMMS in [87] consists of two types of 1-DOF revolute joints: “rotate” and “pivot” joints, actuated by DC motors in conjunction with harmonic drive mechanism, and links of circular cross-section. The mechanical coupling is accomplished using V-band flanges which are an integral part of the link and joint modules. A multiplexed communication link, similar to Local Area Network (LAN), is employed for bidirectional data transmission between modules. Real-time control programs execute on a dedicated

controller CPU, a single-board computer based on Motorola 68020 and VME bus. This controller CPU performs the necessary real time control of the robot and receives commands from a second *master* CPU which executes the event-driven application program. An algorithm to automatically generate the Denavit-Hartenberg parameters of the RMMS is proposed by Kelmar and Khosla [46]. The inverse kinematics of the RMMS is obtained by a numerical inverse kinematics scheme.

The modular robotic system in [17] employs both revolute and prismatic joints which are actuated by DC motors. The revolute joints use harmonic-drive transmissions and the prismatic joints use ball-screw transmission. The schematic drawing of these joints are shown in Fig. 1.1. The links have a circular cross-section. The connection between modules uses a 45° connecting scheme relative to the module's main axes. Two modules can be connected either in a straight line or in perpendicular direction. Rotating the 45° connector by 90 deg. increments can reconfigure the link into an in-plane or out-of-plane link as shown in Fig. 2.3. A SCARA-configuration and an articulated-configuration robot are demonstrated in Fig. 1.2. A kinematic modeling scheme which maps a set of input/output frame relationship into Denavit-Hartenberg parameters is reported in [5].

The modular robots in [103] consist of a variety of rotational joints driven by AC motors in conjunction with differential gears, and links of square cross-section. The compact joint design allows complicated 2-DOF or 3-DOF rotary motions. A simple male-female fastening device is designed to connect modules in any order quickly and exactly. A data file stores all of the important specification of all available elements. A program which selects modules to be assembled according to the task description is proposed.

The CEBOT conceived in [32] employs a different aspect of modular robots. There are three types of modules in the CEBOT: joint cells, branching cells, and working cells. All cells are identical in dimension. The connection between cells are carried out by a hook type coupling mechanism. A cone-shaped mechanism guides the cells during

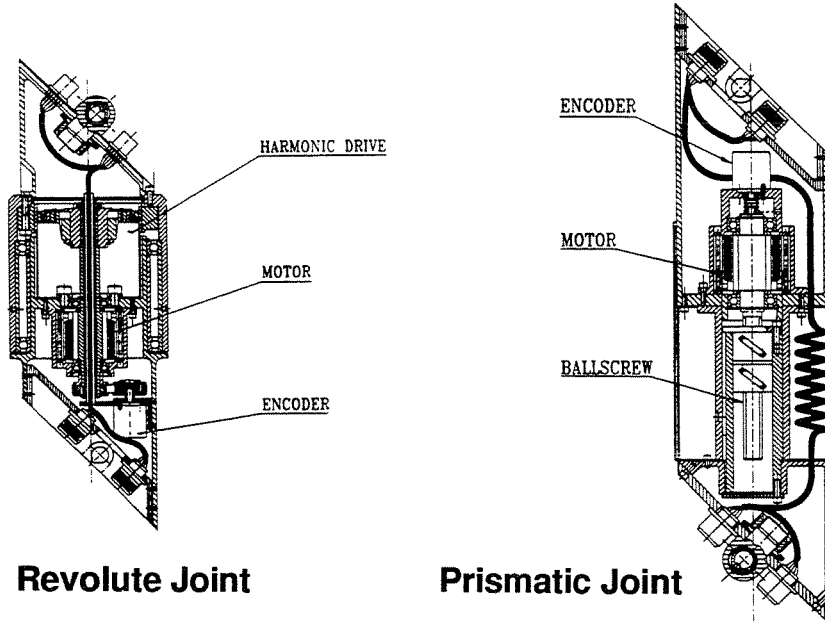


Figure 1.1: Revolute and prismatic joint modules (after [17])

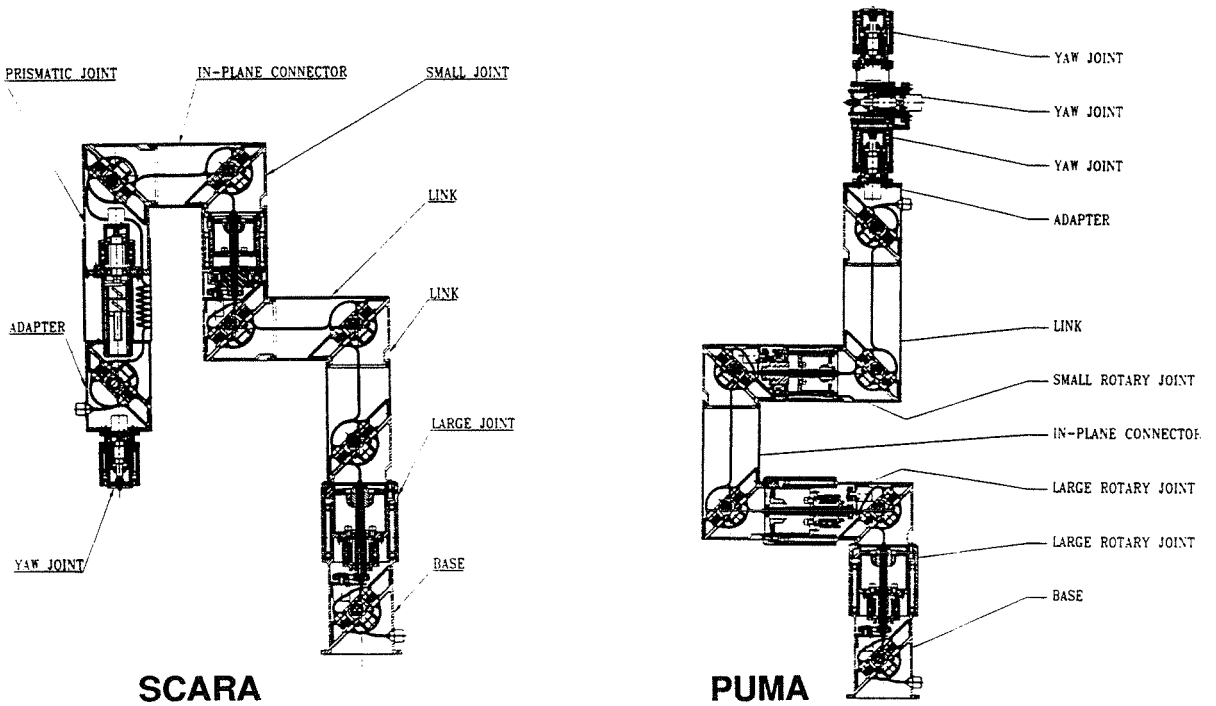


Figure 1.2: Two configurations of UT's modular robots (after [17])

coupling [33]. The cells in the CEBOT possess certain intelligence and communication capability so that they can reconfigure themselves autonomously. With a large number of autonomous cells, the CEBOT becomes a distributed intelligence system. To perform a task collectively, a master cell is selected by using a “network energy,” in which an energy function is calculated based on the information flow in the CEBOT [34]. The master cell becomes the centralized control and communication unit for planning and decomposition of tasks.

1.2. Module Assembly Issues

In the deployment phase of a modular robotic system, a human operator faces the problem of selecting the module assembling sequences and arrangement. Because the modules are designed as standard components that can be assembled in different ways and re-used for different purposes, the operator may want to know how to find an optimal module arrangement for a prescribed robot task from a inventory of system modules. For clarity, the arrangement and assembly of modules in a modular robot is termed an *assembly configuration*. A set of modules can build a number of unique assembly configurations. The module assembly planning problem is to find an optimal one that satisfies the task requirement. Compared to a fixed configuration robot which is usually a compromised design for a set of tasks, a modular robot can accomplish a large class of tasks through reconfiguration of a small inventory of modules.

This module assembly problem has not received sufficient attention in the modular robot literatures. It is treated as an iterative kinematic synthesis problem in [17] and an exhaustive search problem with a predetermined robot topology in [32]. A design grammar approach of enumerating assembly configurations is discussed in [98]. For simple module design, these approaches may apply. As the module design become more complicated, i.e., with more versatile connecting schemes and symmetric geometry, more complicated modular robot structure can be built. Hence, modular robot systems can handle more diversified tasks. The complexity of the module assembly problem increases exponentially. For instance, in the system described in [103], there is no

symmetry in the link modules and only very limited configurations can be constructed. But in the University of Toronto's modular robot system, the links are designed as circular cylinders with symmetry that can be connected to other modules at either end. This symmetry increases the number of possible robot configurations.

In this thesis, the module assembly problem is solved in a systematic way by decomposing it into two separate sub-problems: 1) how to find all possible unique assembly configurations from an inventory of modules; 2) how to determine a sufficient or optimal one among the set of assembly configurations that satisfies a task requirement.

The first sub-problem is termed *module assembly enumeration*, which is basically an algebraic problem involving representation of modular robot assembly configurations, equivalence relations, and counting theorems. The solution to this problem was first reported in Chen and Burdick [14]. An explicit expression of the robot assembly configuration using a graph matrix representation termed an *assembly incidence matrix* (AIM) is defined. This representation was not conceived in any system mentioned in the previous section. For link modules with symmetry, joints can be attached to it in many ways but with identical kinematic properties. Equivalence relation and counting theorem are applied to classify and list distinct assembly patterns. Enumerating distinct robot assembly configuration from an inventory of modules can be done by using the AIM representation and distinct joint assembly patterns. It will be shown in Chapter 3 that this enumeration procedure reduces the complexity and eliminates the generation of a large number of identical assembly configurations by the brute force enumeration scheme mentioned in [32].

The second sub-problem is termed the *task-oriented optimal configurations* which is a combinatorial optimization problem. Kinematic models of a modular robot is developed. It is the basis for robot task evaluation. Because modular robots have no predetermined assembly configuration, in order to retain the modularity of the system, an algorithm to automatically generate robot kinematics from an AIM is introduced in advance. Task requirements and kinematic constraints that affect task execution

are explicitly formulated in a task-oriented objective function. The domain of this objective function is the set of distinct assembly configurations generated from the enumeration procedure. Since the domain is a finite set, combinatorial optimization techniques are employed.

A schematic diagram that depicts the process to solve this module assembly problem is shown in Fig. 1.3. Three blocks show the core of this thesis: module assembly enumerations, modular robot kinematics, and task-oriented optimal configurations.

Lastly, we discuss the application of modular robots to multifinger grasping and manipulation. In order to obtain a stable grasp on an object, a force-closure grasp is necessary. A force-closure grasp is a set of finger contacts on the object surface such that any disturbance force/moment exerted on the object can be balanced by the finger contact forces. For two-finger grasps, antipodal point grasps guarantee force-closure. For n -finger grasps, a force-closure test function is defined on a planar object to check the force-closure condition of a grasp.

1.3. Organization of This Work

The remaining chapters of this thesis are partitioned into three parts: Module models and mathematics pertaining to module assembly problems—Chapters 2-3; Assembly configuration enumeration and task-optimal configuration—Chapters 4-6; Application to multifinger grasping and manipulation—Chapter 7.

Chapter 2 introduces a set of conceptual module models. Two types of modules (joint and link modules) which are related to the regional structure of the robot are considered. A set of basic requirements are defined on joint and link modules. Several types of joints and link are introduced. An example is demonstrated for the transformation between a real module assembly and the modeled one.

Chapter 3 reviews a set of mathematical tools for the module assembly problem including combinatorics and graph theory. Link modules are modeled as symmetric polyhedral objects with multiple *connecting ports*. The rotations of a symmetric link module

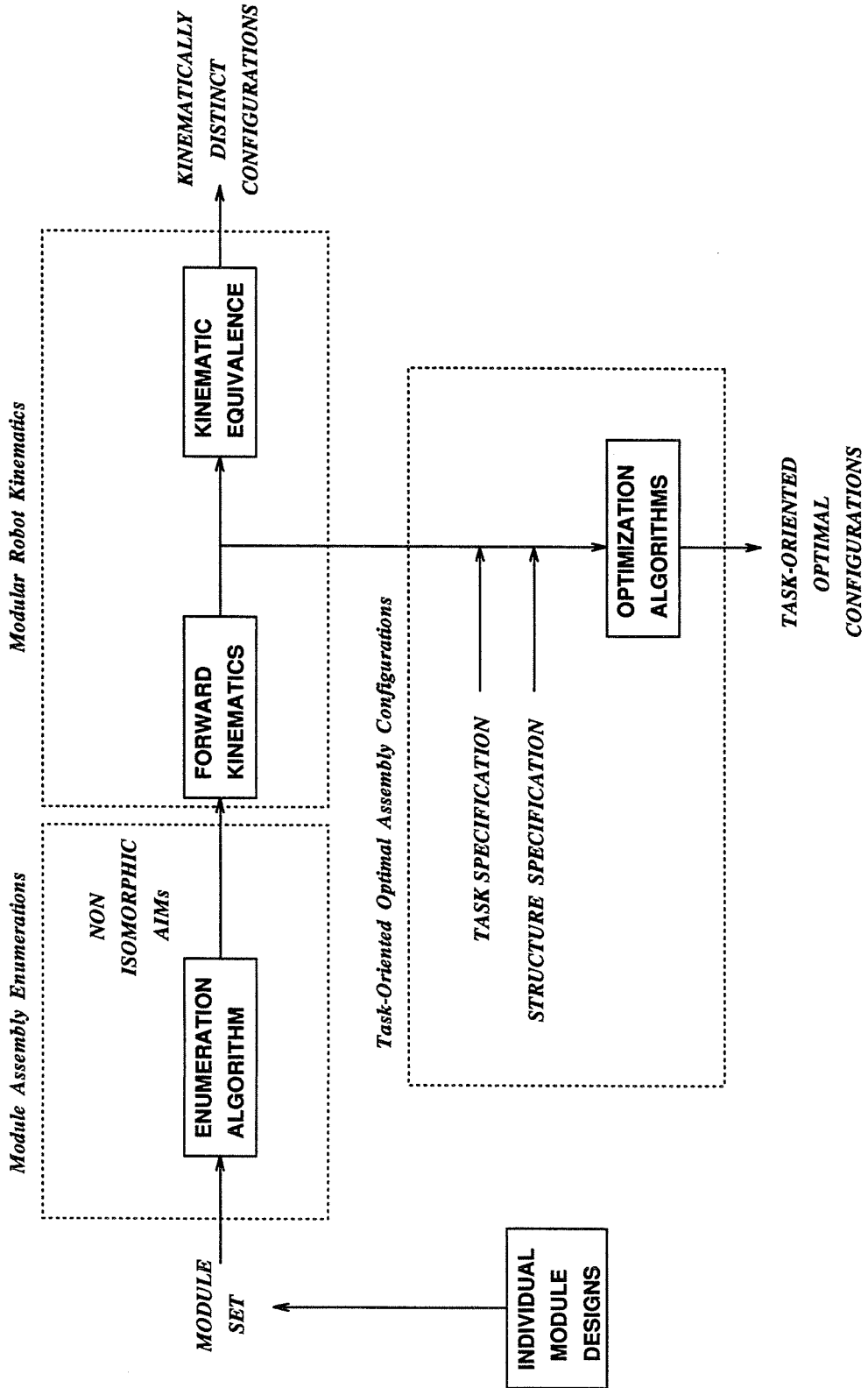


Figure 1.4: Schematic diagram for module assembly problem

are related to permutations on its connecting ports. Kinematic graph techniques are introduced to represent the underlying topology of the modular robot structure.

Chapter 4 presents a methodology to enumerate modular robot assembly configuration. An *assembly incidence matrix* (AIM) is defined for every robot assembly configuration. Equivalence relations based on symmetric rotations of link modules and graph isomorphisms are defined on assembly incidence matrices to classify distinct module assembly. An algorithm based on this equivalence relation to systematically enumerate tree-structured modular robot assembly configurations from a given inventory of modules is proposed. The computation complexity of this algorithm is also discussed. Issues regarding closed-loop module assembly are briefly mentioned.

Chapter 5 studies kinematics issues in modular robots. A product-of-exponentials based kinematic modeling technique is proposed to generate tree-structure robot forward kinematics from a given AIM automatically. A numerical inverse kinematics technique based on Newton-Ralphson method is employed for calculating the inverse kinematic solution. Kinematic equivalence relation is defined on distinct AIMs that possess identical kinematic properties. Modular robots with kinematic equivalent AIMs can perform identical tasks.

Chapter 6 discusses task-optimal assembly configuration problem. The problem is formulated as a combinatorial optimization problem by introducing a task-related objective function. Task and module assembly specifications are explained in detail. They are taken as input parameters to this objective function. The function then evaluates the task performance of an AIM of a modular robot. Genetic algorithms (GA) are proposed for this combinatorial optimization problem. Examples of finding task-optimal serial R-joint robot configurations utilizing GAs are demonstrated.

Chapter 7 develops the application of modular robots to planning multifinger grasps and manipulations. Force-closure conditions are identified on planar and spatial objects with friction contact models. Planning two-finger grasps is done through a set of antipodal point grasps on a smooth shaped object. The antipodal points are found

by solving a constraint global optimization problem on a “grasping energy function” defined at two finger contacts on the object. Planning n-finger grasps, where n is greater than two, is achieved by defining a qualitative force-closure test function on all the n-finger grasps on an object. The application of this test to dextrous finger manipulation and finger gaiting are demonstrated.

Chapter 8 is the conclusion. This contains a discussion of further problems which must be resolved in modular robots and their future development.

Chapter 2

Conceptual Model of Modules

This chapter introduces a conceptual model of modular robot regional structures. The regional structure, which is crucial to the performance of the entire robot system, consists of link and joint modules. By rearranging the assembly sequence of the same set of the modules, one can obtain a set of robot configurations with various parameters which have different kinematic and dynamic behavior. The types of joints determine the final motion of the end-effector. The lengths of the links affect the shape and size of workspaces and singular positions.

The mechanical design of modules varies from system to system. This work is focused on the combinatorial nature of the module assembly problem and on the task performance problem besides modular mechanical constructions. Thus, no specific set of modules will be designed. Instead, a conceptual and abstract module model set based on the features found in many real implementations will be introduced. The modular systems developed or proposed to date have several common mechanical and structural features: (1) simple joint designs: only 1-DOF revolute and 1-DOF prismatic joints are considered [17,32,87]; (2) symmetric link geometries for interchangeability [17,32, 87]; and (3) multiple connection method on a link [17].

The joint modules are assumed to be self-actuated 1- or 2-DOF joints. These joints can create rotary or translational motions. The link modules are polyhedral objects

with geometric symmetry, and multiple connection of joint modules are allowed.

This chapter is organized as follows. Section 2.1 discusses the requirement on joint modules. Section 2.2 introduces the features of link modules. An example is shown to explain the transformation between a real link assembly and the abstract one. Section 2.3 defines several crucial module dimensions to be used in modular robot kinematics.

2.1. Joint Modules

For many robot designs the joint actuators are located close to the base and gear trains or tendons are used to transmit actuator forces. This design strategy reduces the inertia of the entire robot arm. In modular robots, joints have to be self-actuated. It is very inconvenient to put the joint and its actuator in a different location while maintaining the modularity of the entire system. Therefore, a modular robot joint module is not only a “kinematic” joint that connects two rigid bodies, but is also an actuator which generates motion between two adjacent links.

The actual design of the joint modules involves a motor (actuator) and connecting interfaces (control, communication, and mechanical), which is not the topic here; instead, we concentrate our attention on the types of joints creating different motions. There are six types of joints commonly used, termed “lower pair” joints: Revolute joint (1 DOF), Prismatic joint (1 DOF), Screw joint (1 DOF), Cylindrical joint (2 DOF), Planar joint (2 DOF), and Spherical joint (3 DOF) [43]. Combining several 1 DOF joints can create joint motion equivalent to that of a multi-DOF joint. Hence, there is no need to employ complex multi-DOF joints in a modular robot.

The following types of joint modules are assumed to be incorporated in our model modular robotic system: (The notations for joint module types are shown in the parentheses.)

- **Revolute joint (R)**: unlimited 1 degree-of-freedom (DOF) rotary motion between two link modules.

- **Prismatic joint (P)**: limited 1 DOF translational motion between two links.
- **Helical joint (H)**: 1 DOF twisting motion between two links.
- **Cylindrical joint (C)**: 2-DOF motion between connected links: one is rotation, the other is translation along the rotation axis.

All joints are connected to link modules through standardized connecting interfaces. A schematic diagram of these joint modules is shown in Fig. 2.1

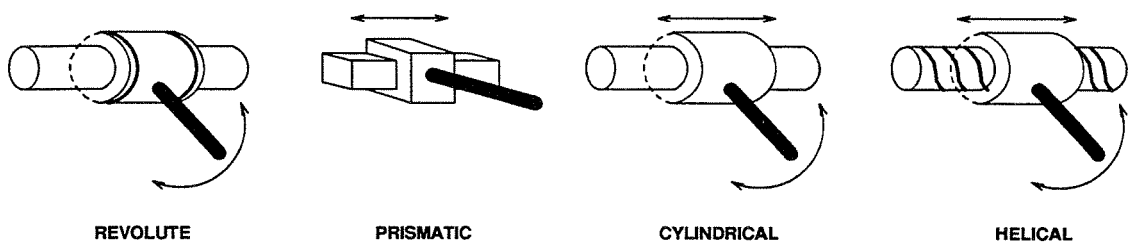


Figure 2.1: Types of joints

2.2. Link Modules

In contrast to joint modules, a link module has no moving parts. It serves solely to connect and support joints. The length of the link between connected joints and the orientations of the joint axes determine the size and shape of the working space of the robot. For versatility, we require that the link modules has two features: (1) multiple joint connections, and (2) symmetry in geometric shape.

Joint modules are connected to the link through *connecting ports*. Changing the number of connected joints alters the function of the link module. For example, with two joints, the link module serves as an ordinary link for the serial manipulator, while with three joints, the link module becomes a branching unit, such as those used in the CEBOT system [32]. Furthermore, changing the location of the joint alters the link parameters such as link lengths, twist angles, and link offsets commonly used in

D-H parameterization of robot kinematics [19]. With multiple choices of connecting ports, the link parameters of the link module has a number of different combinations; therefore, the kinematics of the modular robot changes as well.

Since joints can be attached to a link module in multiple ways, we further assume that the connecting ports are located symmetrically on a link module. The symmetry design allows link modules to be interchanged without any orientation problem and to be re-used easily in different purposes. Although link designs in some real systems do not have symmetry [103], it will be shown later in this thesis that the process of finding joint assembly patterns on a symmetric link module can be useful for unsymmetric links as well. For convenience, a body coordinate system, termed the *module coordinate frame*, is defined on every module whose origin is located at the link's center of symmetry. The connecting ports are labeled accordingly.

For illustration purpose, we assume only two types of link modules are available: square prisms and cubic box units. Other symmetrically shaped objects can be similarly treated.

- **Square prism (L):** This is a prism with a square longitudinal cross section. The ten connecting ports are located symmetrically on each face and are marked from one to ten, as shown in Fig. 2.2. At most, ten different joints can be simultaneously attached to the link. The origin of the module frame is located at the prism's center of symmetry. The z-axis is directed along the prism's primary axis, while the x-axis is perpendicular to one of the faces.
- **Cubic box (B):** This link module has one connector on each of its six faces. The origin of module frame is located at the center of the cube. Its x, y, and z-axes and the port numbering are shown in Fig. 2.2. If six R-joints are attached to the cube simultaneously, their joint axes will be intersected at one point—the centroid of the cube.

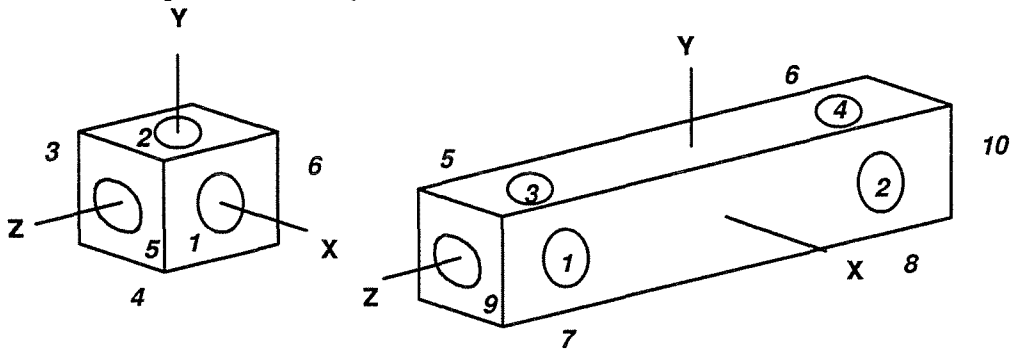


Figure 2.2: Link modules—a cubic box and a prism

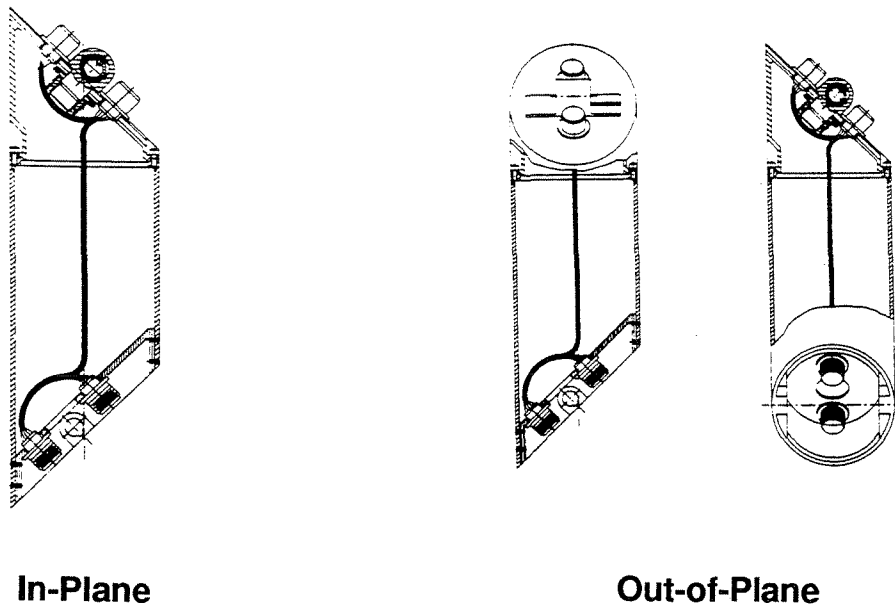


Figure 2.3: Real implementation of UT's connectors

Modelling a link module in a real system is demonstrated in the following example. The in-plane and out-of-plane connectors developed in [17] are shown in Fig. 2.3. Both connectors have circular cross sections. The connecting interfaces between joint modules and the connector are both 45° cross section relative to the longitudinal direction. When two revolute joints are connected to an in-plane connector, their joint axes are in a plane. This corresponds to attaching two R-joints to ports on opposite faces of the prismatic link as shown in Fig. 2.4. When joints are attached to the out-of-plane connector, their joint axes lie in two perpendicular planes, which can be realized by connecting two R-joints on two adjacent faces of the prism shown in the figure.

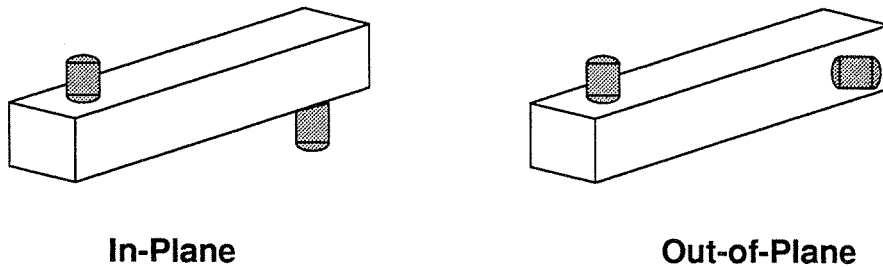


Figure 2.4: Module models of the connectors

2.3. Module Dimensions

Module dimensions will be needed in the discussion of robot kinematics and related task-optimal configuration problem. Without knowing those dimensions kinematic equations cannot be derived. Several crucial module dimensions are defined as follows. Modules of the same type with different dimension are treated like different types of modules.

- Joint modules (Figure 2.5)
 - JointLength: This quantity is defined to be the length of the joint exposed after it is attached to the links, i.e., the distance between the faces of the two links. JointLength of P, H, C-joints whose longitudinal length may vary are defined at their zero positions.
- Square prisms (Figure 2.6)
 - LinkLength: The longitudinal length of the prism. This quantity will be useful when joints are attached to the end face of the prism.
 - LinkWidth: The side length of the end-square.
 - PortSeperation: The distance between the centers of the two connecting ports on the same side-face. If two joints are connected to the ports on the same side, this quantity becomes the actual link length.
- Cubic boxes (Figure 2.6)
 - LinkHeight: The length of the edge of the cube.

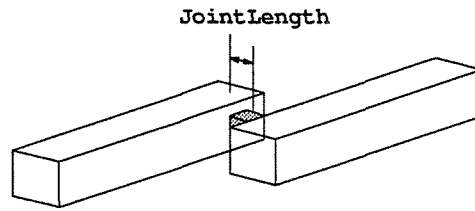


Figure 2.5: Joint dimension

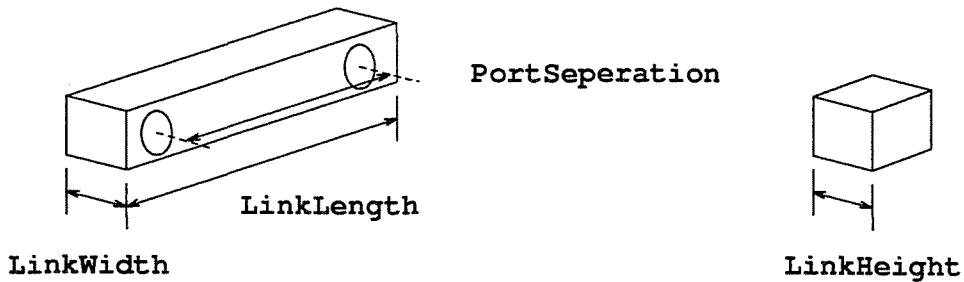


Figure 2.6: Link dimensions

2.4. Discussion

This chapter introduced a set of conceptual link and joint module models which have the characteristics found in existing or foreseeable implementations. The joints and links comprise the regional structure of a modular robot. The regional structure determines the kinematic and dynamic performance of the robot. The joint modules are modeled as self-actuated joints with 1 or 2 DOF. The link modules are modeled as symmetric objects with multiple connecting ports. This modeling technique was demonstrated by an example converting the in-plane and out-of-plane connectors developed by University of Toronto to our module models.

Chapter 3

Mathematics for Modular Robotics

In this chapter, we review and extend a set of basic terminologies in algebra, combinatorics, and graph theory for module robot assembly problem. Owing to geometric symmetry in link modules, group theory and combinatorics will play an important role in counting the distinct assemblies on a single link module, which are the basic building blocks for a large robot structure. The large structure of a modular robot, i.e., the connections between link and joint modules, can be described by a graph, especially, by a *kinematic graph*. It will be shown in the next chapter that by adding more features to this graph representation, the entire robot assembly can be fully represented.

The organization of this chapter is as follows. Section 3.1 reviews several useful concepts in algebra: equivalence classes, groups, and permutations. Section 3.2 describes a class of rotations exhibited in an object due to their symmetry geometry and shows their relation with permutations. Section 3.3 discusses how to count distinct patterns on a symmetric object using Pólya's theorem. Section 3.4 reviews several frequently used definitions in graph theory. Section 3.5 introduces the concept of kinematic graphs for modular robot assemblies.

3.1. Equivalence, Groups, and Permutations

Some of the basic notations and terminologies in algebra that are used throughout this thesis are introduced in this section. For more detail, please refer to [30,39,64].

Let $\mathcal{X} = \{a, b, c, \dots\}$ be a set with a finite number of elements.

Definition 3.1: The binary relation \sim on \mathcal{X} is an *equivalence relation* on \mathcal{X} such that for all $a, b, c \in \mathcal{X}$,

1. $a \sim a$; (reflexivity)
2. if $a \sim b$, then $b \sim a$; (symmetry)
3. if $a \sim b$ and $b \sim c$, then $a \sim c$. (transitivity)

Definition 3.2: If \sim is an equivalence relation on \mathcal{X} , the *equivalence class* of $a \in \mathcal{X}$ is the set defined by

$$[a] = \{x \in \mathcal{X} | x \sim a\} \quad (3.1)$$

Note that \mathcal{X} is the disjoint union of its equivalence classes. The set of equivalence classes is denoted by \mathcal{X} / \sim .

Definition 3.3: A *group* is a non-empty set \mathcal{G} , along with a binary operation, $*$, such that for $a, b, c \in \mathcal{G}$,

1. $a * b \in \mathcal{G}$; (closure)
2. $(a * b) * c = a * (b * c)$; (associativity)
3. there exists a unique element $e \in \mathcal{G}$ such that $a * e = e * a = a$ for every $a \in \mathcal{G}$;
(identity)
4. for every $a \in \mathcal{G}$, there exists a unique $a^{-1} \in \mathcal{G}$ such that $a * a^{-1} = a^{-1} * a = e$.
(inverse)

Definition 3.4: [30] Let \mathcal{X} be a set and \mathcal{G} a group. An *action* of \mathcal{G} on \mathcal{X} is a map, $\star: \mathcal{X} \times \mathcal{G} \rightarrow \mathcal{X}$, such that for all $x \in \mathcal{X}$

1. $x \star e = x$, where e is the identity element of \mathcal{G} ;
2. $x \star (g_1 \circ g_2) = (x \star g_1) \star g_2$, where $g_1, g_2 \in \mathcal{G}$.

\mathcal{X} is called a \mathcal{G} -set.

Theorem 3.5: [30] Let \mathcal{X} be a \mathcal{G} -set. $x_1, x_2 \in \mathcal{X}$ are said to be equivalent, i.e., $x_1 \sim x_2$ iff there exists a $g \in \mathcal{G}$ such that $x_1 \star g = x_2$.

Definition 3.6: The equivalence class of $x \in \mathcal{X}$ induced by \sim is defined by

$$[x] = \{y \in \mathcal{X} | y = x \star g, \exists g \in \mathcal{G}\} \equiv x \star \mathcal{G}. \quad (3.2)$$

It is called an orbit of \mathcal{X} under the action of \mathcal{G} .

Definition 3.7: Let $\mathcal{X} = \{x_1, \dots, x_n\}$ be a finite set with n elements. A permutation π on \mathcal{X} is a 1-to-1 mapping of \mathcal{X} onto itself written as follows.

$$\pi = \begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_{i_1} & x_{i_2} & x_{i_3} & \cdots & x_{i_n} \end{pmatrix}, \quad (3.3)$$

where x_{i_k} is the image of x_k under π and $i_k \in \mathcal{I} = \{1, \dots, n\}$. \mathcal{I} is called the *index set*. In short, we write π as the permutation of the index set.

$$\pi = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ i_1 & i_2 & i_3 & \cdots & i_n \end{pmatrix}, \quad (3.4)$$

Example 3.8: A permutation π on a set of 4 elements can be written as

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}, \quad (3.5)$$

where i represents the first element x_i and $i \in \mathcal{I} = \{1, 2, 3, 4\}$.

Suppose π_1 and π_2 are permutations on \mathcal{X} and $\pi_1 \neq \pi_2$. The composition of the two, $\pi_1 \circ \pi_2$, is also a permutation on \mathcal{X} . If π_3 is also a permutation on \mathcal{X} , one can show that $(\pi_1 \circ \pi_2) \circ \pi_3 = \pi_1 \circ (\pi_2 \circ \pi_3)$. The identity permutation, e , will look like

$$\pi = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ 1 & 2 & 3 & \cdots & n \end{pmatrix}, \quad (3.6)$$

and we can always find an inverse permutation of π_1 such that $\pi_1^{-1} \circ \pi = e$. Therefore, a set of permutations on \mathcal{X} with the associated operator, \circ , is called a *permutation group* on \mathcal{X} and is denoted by $\mathcal{S}_{\mathcal{X}}$ if it satisfies Definition 3.3.

3.2. Symmetric Rotations

Objects with geometric symmetry like square prisms and cubes exhibit a class of body rotations which preserve the orientations and positions of their shape. Such rotations are termed *symmetric rotations*, since one cannot distinguish between the rotated and unrotated states of the object. For example, an unmarked square prism which is rotated about its longitudinal axis by 90° , 180° , or 270° looks exactly the same.

Definition 3.9: Let a symmetric object, \mathcal{L} , be a collection of points in \mathbb{R}^3 and let the center of symmetry coincide with the origin of \mathbb{R}^3 . A *symmetric rotation*, $\varphi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, of \mathcal{L} is an element of $SO(3)$ that maps \mathcal{L} to itself, i.e., $\varphi(\mathcal{L}) = \mathcal{L}$.

One can show that the symmetric rotations of an object form a group called the *symmetric rotation group* \mathcal{R} [30,64], where

$$\mathcal{R} = \{\varphi \in SO(3) | \varphi(\mathcal{L}) = \mathcal{L}\}. \quad (3.7)$$

Note that the number of elements in \mathcal{R} of a symmetric polyhedron is finite. Objects of revolution, such as a sphere or cylinder, have infinite, or continuous, rotation groups. Here we only consider finite symmetric rotation groups.

Recall that link modules are assumed to be symmetric objects like cubes and square prisms which have symmetric rotations. In order to identify these rotations, one has to put features on these objects. The following theorem allows us to categorize symmetric rotations in an object.

Theorem 3.10: *Cayley's Theorem* [39]

Every group with finite elements is isomorphic to a subgroup of a permutation group on a set of n elements for some integer n .

Also recall that every link module connecting port is assigned a unique index and these ports are located symmetrically on the link. While a symmetric rotation $\varphi \in \mathcal{R}$ does not alter the final appearance of the link, it does change the connecting port locations. One can imagine that the port locations after the rotation are a permutation of the indices before the rotation operation. As shown in Fig. 3.1, the rotation of the prism link module about its z-axis by 90° causes port 1 to move to where port 3 was, port 2 to port 4, port 7 to port 1, etc. Port 9 and 10 remain the same. This action can be written as a permutation

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 3 & 4 & 5 & 6 & 7 & 8 & 1 & 2 & 9 & 10 \end{pmatrix}, \quad (3.8)$$

or in short, $\pi = (3, 4, 5, 6, 7, 8, 1, 2, 9, 10)$.

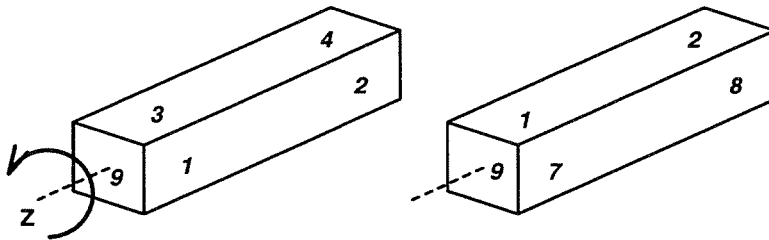


Figure 3.1: Symmetric rotation about z-axis by 90°

Suppose there are n ports on a link and they are labeled from 1 to n . Let $\text{PORT} = \{1, \dots, n\}$ be a set of indices containing all port numbers on a link module. A symmetric rotation $\varphi \in \mathcal{R}$ corresponds to a permutation on PORT . It can be shown that all permutations which correspond to symmetric rotations on a link form a permutation group on PORT denoted by \mathcal{S} . By Theorem 3.10, the symmetric rotation group \mathcal{R} is isomorphic to the permutation group \mathcal{S} on PORT . Hereafter, we use either $\pi \in \mathcal{S}$ or $\varphi \in \mathcal{R}$ to represent a symmetric rotation on the link module.

Let $\text{PORT}(L) = \{1, 2, \dots, 10\}$ and $\text{PORT}(B) = \{1, 2, \dots, 6\}$ be the sets of port indices on a square prism and a cube respectively. Denoting \mathcal{R}_L and \mathcal{R}_B as the symmetric rotation groups on prismatic and cubic links, and \mathcal{S}_L and \mathcal{S}_B as the permutation groups

on $\text{PORT}(L)$ and $\text{PORT}(B)$, Theorem 3.10 says \mathcal{R}_L (\mathcal{R}_B) is isomorphic to \mathcal{S}_L (\mathcal{S}_B). Tables 3.1 and 3.2 list all symmetric rotations and corresponding permutations of the prism and the cube.

| Rotations | | Permutations on I_L | | | | | | | | | | |
|--------------|-------|-----------------------|---|---|---|---|---|---|---|----|----|---------------------------------|
| Axis | Angle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Type |
| identity | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | {10, 0, 0, 0, 0, 0, 0, 0, 0, 0} |
| k | 90° | 3 | 4 | 5 | 6 | 7 | 8 | 1 | 2 | 9 | 10 | {2, 0, 0, 2, 0, 0, 0, 0, 0, 0} |
| k | 180° | 5 | 6 | 7 | 8 | 1 | 2 | 3 | 4 | 9 | 10 | {2, 0, 0, 4, 0, 0, 0, 0, 0, 0} |
| k | 270° | 7 | 8 | 1 | 2 | 3 | 4 | 5 | 6 | 9 | 10 | {2, 0, 0, 2, 0, 0, 0, 0, 0, 0} |
| i | 180° | 2 | 1 | 8 | 7 | 6 | 5 | 4 | 3 | 10 | 9 | {0, 5, 0, 0, 0, 0, 0, 0, 0, 0} |
| j | 180° | 6 | 5 | 4 | 3 | 2 | 1 | 8 | 7 | 10 | 9 | {0, 5, 0, 0, 0, 0, 0, 0, 0, 0} |
| i + j | 180° | 4 | 3 | 2 | 1 | 8 | 7 | 6 | 5 | 10 | 9 | {0, 5, 0, 0, 0, 0, 0, 0, 0, 0} |
| i - j | 180° | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 10 | 9 | {0, 5, 0, 0, 0, 0, 0, 0, 0, 0} |

Table 3.1: Symmetric rotation group \mathcal{R}_L of a prism L

Note that the representation of a symmetric rotation group depends on the features on the object; therefore, is not unique. Here a permutation of the indices on the connecting ports is chosen to represent a rotation. The same symmetric rotation group can be expressed by different permutation groups by adding more features to the symmetric object. Only the number of elements in these groups remains identical. If an object has no symmetry, its symmetric rotation group contains only one element—the identity rotation.

For module assembly problem, the locations of connecting ports are directly related to the kinematics of the robot; hence, they are more important than the shape of the link. The assumption on symmetry of link shapes can be relaxed. However, locations of the connecting ports must remain symmetrical.

3.3. Pólya's Counting Theorem

First let us consider a cube coloring problem. Suppose the six faces of a cube are to be painted with two different colors, say black and white. Each face is allowed to

| Rotations | | Permutations on I_B | | | | | | Type |
|------------------|-------|-----------------------|---|---|---|---|---|--------------------|
| Axis | Angle | 1 | 2 | 3 | 4 | 5 | 6 | |
| identity | | 1 | 2 | 3 | 4 | 5 | 6 | {6, 0, 0, 0, 0, 0} |
| k | 90° | 2 | 3 | 4 | 1 | 5 | 6 | {2, 0, 0, 1, 0, 0} |
| k | 180° | 3 | 4 | 1 | 2 | 5 | 6 | {2, 2, 0, 0, 0, 0} |
| k | 270° | 4 | 1 | 2 | 3 | 5 | 6 | {2, 0, 0, 1, 0, 0} |
| i | 90° | 1 | 5 | 3 | 6 | 4 | 2 | {2, 0, 0, 1, 0, 0} |
| i | 180° | 1 | 4 | 3 | 2 | 6 | 5 | {2, 2, 0, 0, 0, 0} |
| i | 270° | 1 | 6 | 3 | 5 | 2 | 4 | {2, 0, 0, 1, 0, 0} |
| j | 90° | 5 | 2 | 6 | 4 | 3 | 1 | {2, 0, 0, 1, 0, 0} |
| j | 180° | 3 | 2 | 1 | 4 | 6 | 5 | {2, 2, 0, 0, 0, 0} |
| j | 270° | 6 | 2 | 5 | 4 | 1 | 3 | {2, 0, 0, 1, 0, 0} |
| i + j | 180° | 2 | 1 | 4 | 3 | 6 | 5 | {0, 3, 0, 0, 0, 0} |
| i - j | 180° | 4 | 3 | 2 | 1 | 6 | 5 | {0, 3, 0, 0, 0, 0} |
| j + k | 180° | 3 | 5 | 1 | 6 | 2 | 4 | {0, 3, 0, 0, 0, 0} |
| j - k | 180° | 3 | 6 | 1 | 5 | 4 | 2 | {0, 3, 0, 0, 0, 0} |
| i + k | 180° | 5 | 4 | 6 | 2 | 1 | 3 | {0, 3, 0, 0, 0, 0} |
| i - k | 180° | 6 | 4 | 5 | 2 | 3 | 1 | {0, 3, 0, 0, 0, 0} |
| i + j + k | 120° | 2 | 5 | 4 | 6 | 1 | 3 | {0, 0, 2, 0, 0, 0} |
| i + j + k | 240° | 5 | 1 | 6 | 3 | 2 | 4 | {0, 0, 2, 0, 0, 0} |
| i + j - k | 120° | 6 | 1 | 5 | 3 | 4 | 2 | {0, 0, 2, 0, 0, 0} |
| i + j - k | 240° | 2 | 6 | 4 | 5 | 3 | 1 | {0, 0, 2, 0, 0, 0} |
| j + k - i | 120° | 6 | 3 | 5 | 1 | 2 | 4 | {0, 0, 2, 0, 0, 0} |
| j + k - i | 240° | 4 | 5 | 2 | 6 | 3 | 1 | {0, 0, 2, 0, 0, 0} |
| j - k - i | 120° | 4 | 6 | 2 | 5 | 1 | 3 | {0, 0, 2, 0, 0, 0} |
| j - k - i | 240° | 5 | 3 | 6 | 1 | 4 | 2 | {0, 0, 2, 0, 0, 0} |

Table 3.2: Symmetric rotation group \mathcal{R}_B of a cube B

be painted either black or white, and the six faces can be all black or all white. We would like to find a method to count all possible ways of painting this cube. A brute force approach to do that is to mark the six faces and find different color combinations among them. The result is $2^6 = 64$ ways of painting. Since the cube is a symmetric object, a lot of these patterns look identical after a symmetric rotation. The actual number of distinct patterns will be less than 64.

For this particular problem, the number of distinct patterns can still be found by hand. When the number of colors increases or the object becomes a polyhedron, the number of distinct painting patterns increases exponentially. Thus, Pólya's counting theorem

provides us a closed form solution to find these distinct patterns in an efficient way. Before the introduction of this theorem, one more definition is introduced.

A permutation, π , on index set, \mathcal{I} , splits the index set uniquely into disjoint subsets called *cycles*, which contain elements of \mathcal{I} cyclically permuted by π . A cycle is of length m if $\pi^m(s) = s \in \mathcal{I}$ and $s, \pi(s), \dots, \pi^{m-1}(s)$ are all contained in this cycle. Let $n = |\mathcal{I}|$, the total number of elements. We say π is of type $\{b_1, b_2, \dots, b_n\}$ if it splits \mathcal{I} into b_1 cycles of length 1, b_2 cycles of length 2, and so on. Note that $b_i = 0$ for $i > n$ and $b_1 + 2b_2 + \dots + nb_n = n$. For example, consider a permutation π_1 on $\text{PORT}(L)$ which corresponds to the rotation of $+90^\circ$ about $z(\mathbf{k})$ axis in Table 3.1. It splits $\text{PORT}(L)$ into 4 cycles: $\{1, 3, 5, 7\}$, $\{2, 4, 6, 8\}$, $\{9\}$, and $\{10\}$, and one can verify that $\{1, 3, 5, 7\} = \{1, \pi_1(1), \pi_1^2(1), \pi_1^3(1)\}$, etc. Therefore, π_1 is of type $\{2, 0, 0, 2, 0, 0, 0, 0, 0, 0\}$.

Definition 3.11: *Cycle Index [4,59,100]*

Let \mathcal{S} be a permutation group on \mathcal{I} . The *cycle index* of \mathcal{S} is defined to be a polynomial in n dummy variables x_1, \dots, x_n as follows. For each $\pi \in \mathcal{S}$ of type $\{b_1, \dots, b_n\}$ forms a product $x_1^{b_1} x_2^{b_2} \dots x_n^{b_n}$ termed the *cycle structure* of π . The cycle index, $P_{\mathcal{S}}$, is the sum of these terms divided by the number of elements in \mathcal{S} , i.e.,

$$P_{\mathcal{S}}(x_1, \dots, x_n) = \frac{1}{|\mathcal{S}|} \sum_{\pi \in \mathcal{S}} x_1^{b_1} x_2^{b_2} \dots x_n^{b_n}. \quad (3.9)$$

For example, the cycle index of the permutation groups on $\text{PORT}(L)$ and $\text{PORT}(B)$ are

$$P_{\mathcal{S}_L}(x_1, x_2, \dots, x_{10}) = \frac{1}{8}(x_1^{10} + x_1^2 x_2^4 + 2x_1^2 x_4^2 + 4x_2^5) \quad (3.10)$$

$$P_{\mathcal{S}_B}(x_1, x_2, \dots, x_6) = \frac{1}{24}(x_1^6 + 3x_1^2 x_2^2 + 6x_1^2 x_4 + 6x_2^3 + 8x_3^2). \quad (3.11)$$

Let $\mathcal{I} = \{1, \dots, n\}$ and $\mathcal{X} = \{a_1, \dots, a_m\}$ be two sets, and let \mathcal{S} be a permutation group of \mathcal{I} . Suppose f is an injection mapping from \mathcal{I} to \mathcal{X} called a *pattern*. The set of patterns is denoted by $\mathcal{X}^{\mathcal{I}}$. From Definition 3.4, \mathcal{S} induces an action on the pattern set $\mathcal{X}^{\mathcal{I}}$, and thus defines an equivalence relation on it, since $f \circ \pi = g \in \mathcal{X}^{\mathcal{I}}$ and $f \circ e = f$, where $\pi, e \in \mathcal{S}$ and \circ is the composition operator. Let $[f]$ denote the equivalence class containing the pattern f . Equivalent patterns are in the same equivalence class; distinct

patterns belong to different equivalence classes. The inventory of the equivalence class of the patterns (or pattern inventory) can be found by Pólya’s theorem as follows.

Theorem 3.12: *Pólya’s Counting Theorem [4,59]*

Let P_S be the cycle index of S . Assign a dummy variable y_i to every element in \mathcal{X} , e.g., y_1 to a_1 , y_2 to a_2 , etc. The inventory of equivalence classes, I_P [59], in $\mathcal{X}^{\mathcal{I}}$ can be found by substituting x_k in P_S with $\sum y_i^k$, i.e.,

$$I_P = P_S(\sum y_i, \sum y_i^2, \dots, \sum y_i^n), \tag{3.12}$$

where $n = |\mathcal{I}|$. The coefficient of term $y_1^{d_1} y_2^{d_2} \dots$ in (3.12) indicates the number of distinct patterns (or equivalence classes) with d_1 copies of a_1 ’s, d_2 copies of a_2 ’s, etc.

Example 3.13: Now let’s solve the cube painting problem using Pólya’s theorem. Let $\mathcal{I} = \{1, \dots, 6\}$ represent the six cube faces and $\mathcal{X} = \{B, W\}$, the types of colors. A painted pattern is a function $f : \mathcal{I} \rightarrow \mathcal{X}$. Two example patterns are

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ B & W & B & B & B & B \end{pmatrix}, \quad g = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ B & B & B & B & W & B \end{pmatrix}. \tag{3.13}$$

Face 2 of Pattern f and Face 5 of Pattern g are painted white; other faces are all painted black. f and g are similar patterns because a symmetric rotation can be employed on f to match g . As mentioned earlier, a symmetric rotation on the cube can be expressed as a permutation on the number of faces, $\pi : \mathcal{I} \rightarrow \mathcal{I}$, such that the composition, $f \circ \pi$, becomes another pattern. We can therefore use an equivalence relation induced by the symmetric rotation group of the cube \mathcal{R}_B (or the corresponding permutation group \mathcal{S}_B) to classify these patterns. Two color patterns f_1 and f_2 are said to be equivalent (or look alike) if any only if $f_1 \circ \pi = f_2$, where $\pi \in \mathcal{S}_B$. Similar patterns are in the same equivalence class; distinct patterns are in different equivalence classes. In other words, finding the number of distinct color patterns on a cube is the same as finding the number of equivalence classes in the pattern set $\mathcal{X}^{\mathcal{I}}$ (or all orbits of $\mathcal{X}^{\mathcal{I}}$ under the action of \mathcal{R}_B). Let $[f_1]$ be the equivalence class containing the color pattern f_1 , then $\mathcal{X}^{\mathcal{I}}/\mathcal{R}_B$ represents the set of equivalence classes under the action of \mathcal{R}_B , i.e., the set of distinct color patterns.

Now assign y_1 to B and y_2 to W; the inventory pattern is found by substituting $\sum y_i$, $\sum y_i^2$, $\sum y_i^3$, and $\sum y_i^4$ into (3.11):

$$\begin{aligned} I_P &= P_{S_B}(\sum y_i, \sum y_i^2, \sum y_i^3, \sum y_i^4) \\ &= y_1^6 + y_1^5 y_2 + 2 y_1^4 y_2^2 + 2 y_1^3 y_2^3 + 2 y_1^2 y_2^4 + y_1 y_2^5 + y_2^6 \end{aligned} \quad (3.14)$$

The coefficient of $y_1^5 y_2$ is 1, which means there is one way to paint one face black and five others white. There are two distinct ways to paint three faces white and three faces black as indicated by the coefficient of $y_1^3 y_2^3$. The total number of distinct patterns using only black and white colors is the summation of all coefficients in I_P , which is 10, far less than 64. ■

3.4. Graphs

This section reviews some graph terminologies used in the sequel. For a more complete exposition of graph theory and graph algorithms, please refer to [26,44,73,89].

Basic Graph Definitions

A graph $G = (V, E)$ consists of a vertex set, V , and an edge set, E , such that every edge in E is associated with a pair of vertices. A *labeled* graph is a graph whose vertices are labeled v_1, v_2 , etc., and whose edges are labeled e_1, e_2 , etc., such that $V = \{v_1, \dots, v_m\}$ and $E = \{e_1, \dots, e_n\}$. In a labeled graph, we usually do not differentiate among the types of vertices (edges). If vertices (edges) are assumed to be different, a more complex graph structure can be defined. Let \tilde{V} (\tilde{E}) be the set of types of vertices (edges) on G , and $f_v : V \rightarrow \tilde{V}$ and $f_e : E \rightarrow \tilde{E}$ be injection mappings. The mapping f_v and f_e are called *vertex* and *edge assignments* respectively. The pair (f_v, f_e) is called an *assignment* of G .

Definition 3.14: A specialized graph \tilde{G} is a labeled graph G with an assignment, (f_v, f_e) , on its vertices and edges. It can be written as $\tilde{G} = (G, \tilde{V}, \tilde{E}, f_v, f_e)$.

The graph coloring problem, i.e., the minimum number of colors required to paint vertices of a graph, is a well-studied subject in graph theory. The colored graph can

be considered as a specialized graph in which the set of vertex type \tilde{V} contains colors on the vertices and the set of edge type, \tilde{E} , contains only one element because the edges are not required to be painted and are treated as identical. A specialized graph example is shown in Fig. 3.2(B), where $\tilde{V} = \{R, G, W\}$ and $\tilde{E} = \{a, b, c\}$.

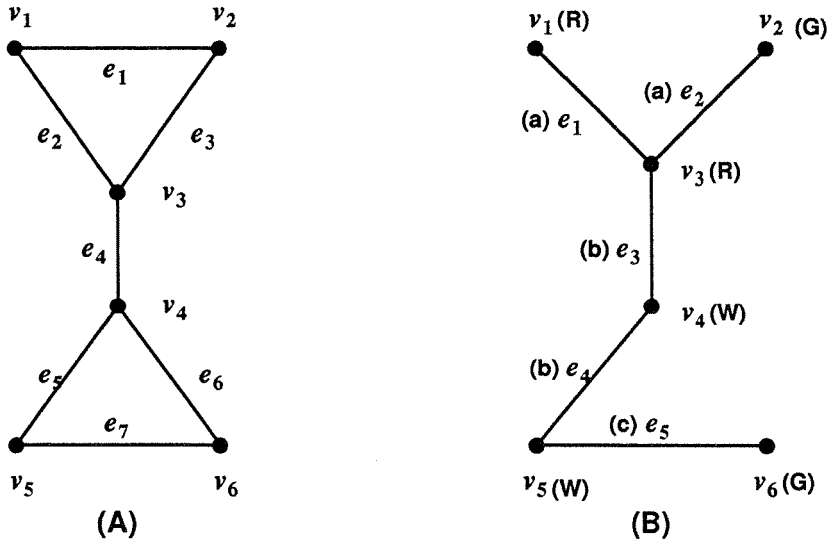


Figure 3.2: A labeled graph and a specialized graph

Incidence Matrices

The topology of a graph can be shown pictorially or in matrix forms. Pictorial representation is good for visualization only; matrix representations are convenient and useful for computer processing. A variety of graph matrix representations have been developed for labeled graphs for different purposes, such as the adjacency matrix or cycle matrix [22,37]. In order to fully describe the incidence relationship among all vertices and edges, a *vertex-edge incidence matrix* representation is chosen.

The incidence matrix, $M(G)$, of a graph G with m vertices and n edges is an $m \times n$ matrix whose entries contains 0's and 1's only. Entry m_{ij} is equal to 1 if edge e_j is incident on vertex v_i ; equal to 0, otherwise. For example, the labeled graph of Fig. 3.2(A) can be written as

$$M(G) = \begin{matrix} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix}. \quad (3.15)$$

A specialized graph can be described in an incidence-matrix-like structure as well. An additional row (column) is required for the assignment of types of vertices (edges). Let \tilde{G} be a specialized graph with m vertices and n edges. The following matrix representation is defined for \tilde{G} .

Definition 3.15: The *extended incidence matrix* (eIM) of the specialized graph \tilde{G} denoted by $M(\tilde{G})$ is an $(m+1) \times (n+1)$ matrix such that

1. $m_{ij} = 1$, if e_j is incident on v_i , and $m_{ij} = 0$, otherwise, $i = 1, \dots, m$, $j = 1, \dots, n$.
2. $m_{i,n+1} = f_v(v_i) \in \tilde{V}$, which is v_i 's vertex assignment, $i = 1, \dots, m$.
3. $m_{m+1,j} = f_e(e_j) \in \tilde{E}$, which is e_j 's edge assignment, $j = 1, \dots, n$.
4. $m_{m+1,n+1} = 0$.

The upper-left $m \times n$ submatrix of $M(\tilde{G})$ is identical to the incidence matrix of its underlying labeled graph G because they have identical graph topology. The vertex and edge assignments are kept in the last column and the last row of $M(\tilde{G})$ respectively. The eIM of the specialized graph in Fig. 3.2(B) is

$$M(\tilde{G}) = \begin{matrix} & e_1 & e_2 & e_3 & e_4 & e_5 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & R \\ 0 & 1 & 0 & 0 & 0 & G \\ 1 & 1 & 1 & 0 & 0 & R \\ 0 & 0 & 1 & 1 & 0 & W \\ 0 & 0 & 0 & 1 & 1 & W \\ 0 & 0 & 0 & 0 & 1 & G \\ a & a & b & b & c & 0 \end{pmatrix} \end{matrix}. \quad (3.16)$$

Graph Isomorphisms

Definition 3.16: Two labeled graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* iff there exists two bijective mapping $\gamma_{v_{12}} : V_1 \rightarrow V_2$ and $\gamma_{e_{12}} : E_1 \rightarrow E_2$. That is, there exists a 1-to-1 correspondence between their vertex and edge sets that conserves the incidence relations. We call $\gamma_{12} = (\gamma_{v_{12}}, \gamma_{e_{12}})$ an isomorphism from G_1 to G_2 .

Note that $\gamma_{21} = \gamma_{12}^{-1}$. If $V_1 = V_2$ and $E_1 = E_2$, $\gamma_{v_{12}} (\gamma_{e_{12}})$ can be thought as permutations on V_1 (E_1), or equivalently, row (column) permutations on $M(G_1)$.

Theorem 3.17: [22] Two graphs are isomorphic iff their incidence matrices differ only by permutations of rows and columns.

Let $M_{\gamma_{12}}(G_1)$ denote the incidence matrix of G_1 after the permutation $\gamma_{12} = (\gamma_{v_{12}}, \gamma_{e_{12}})$. By the above theorem, we have $M_{\gamma_{12}}(G_1) = M(G_2)$. Fig. 3.3 shows three isomorphic graphs. The isomorphism from (a) to (b) is $\gamma_{ab} = ((2, 1, 4, 3), (c, b, a))$; the one from (a) to (c) is $\gamma_{ac} = ((1, 2, 4, 3), (a, c, b))$.

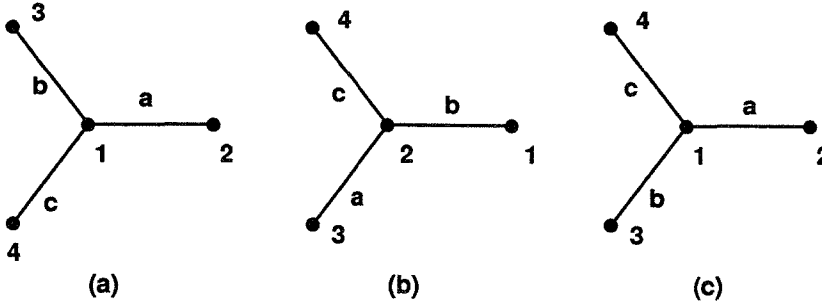


Figure 3.3: Isomorphism and automorphism of graphs

Definition 3.18: Let $\tilde{G}_1 = (G_1, \tilde{V}_1, \tilde{E}_1, f_{v_1}, f_{e_1})$ and $\tilde{G}_2 = (G_2, \tilde{V}_2, \tilde{E}_2, f_{v_2}, f_{e_2})$ be two specialized graphs. \tilde{G}_1 and \tilde{G}_2 are isomorphic iff the following conditions are satisfied:

1. G_1 is isomorphic to G_2 .
2. $\tilde{V}_1 = \tilde{V}_2$ and $\tilde{E}_1 = \tilde{E}_2$.
3. Let $\gamma_{12} = (\gamma_{v_{12}}, \gamma_{e_{12}})$ be the isomorphism from G_1 to G_2 , then $f_{v_1} \circ \gamma_{v_{12}}^{-1} = f_{v_2}$ and $f_{e_1} \circ \gamma_{e_{12}}^{-1} = f_{e_2}$.

The third condition requires that the vertex (edge) assignments differ only by a permutation for isomorphic specialized graphs. Similar to the labeled graph case, the isomorphism γ_{12} can still be thought of as column and row permutations on its eIM $M(\tilde{G}_1)$. The row with vertex assignment and the column of edge assignment also follow the permutation actions. Let $M_{\gamma_{12}}(\tilde{G}_1)$ denote the extended incidence matrix of \tilde{G}_1 after the permutation γ_{12} , then $M_{\gamma_{12}}(\tilde{G}_1) = M(\tilde{G}_2)$.

Graph Automorphisms

If a labeled graph, G , exhibits geometric symmetry, there exists isomorphisms of G to itself, i.e., there exists γ such that $M_\gamma(G) = M(G)$. We call such isomorphism an *automorphism*. For example, γ_{ac} is an automorphism of the graph in Fig. 3.3(a). $M_{\gamma_{ac}}(G_a) = M(G_a)$. One can think of γ_{ac} as a 180° rotation about the edge a of G_a , which transforms G_a into G_c and $M(G_a) = M(G_c)$.

These automorphisms form a group called the *automorphism group* of G , denoted by $\mathcal{H}(G)$. The automorphism group of the graph in Fig. 3.3(a) contains six elements which can be written in permutation group form:

$$\begin{aligned} &((1, 2, 3, 4), (a, b, c)) \quad ((1, 3, 4, 2), (b, c, a)) \quad ((1, 4, 2, 3), (c, a, b)) \\ &((1, 2, 4, 3), (a, c, b)) \quad ((1, 4, 3, 2), (c, b, a)) \quad ((1, 3, 2, 4), (b, a, c)) \end{aligned}$$

where $((1, 2, 3, 4), (a, b, c))$ is the identity element. Note that the automorphism group of an asymmetric graph contains the identity element only.

Similarly, an automorphism, η , of a specialized graph \tilde{G} , is an isomorphism of \tilde{G} to itself which will render $M_\eta(\tilde{G}) = M(\tilde{G})$. \tilde{G} also has an automorphism group $\mathcal{H}(\tilde{G})$ if it exhibits symmetry. Because \tilde{G} has the same underlying graph topology with its labeled graph G , this group is a subgroup of the automorphism group of G , i.e., $\mathcal{H}(\tilde{G}) \subset \mathcal{H}(G)$.

3.5. Kinematic Graphs

In mechanism design, kinematic chains of links and joints are often represented by graphs. The topology of an underlying graph strongly influences the features and

functions of a mechanism. A logical way to convert a kinematic chain to a graph is to replace the joints by edges and links by vertices, since a joint can be connected to two links only and a link may attach many joints. Such graph representations have been called *kinematic graphs* by Freudenstein and Dobrjansky [24], and have been used in mechanism designs for many years. With many well-developed graph tools on hand, e.g., graph enumeration algorithms, graph isomorphism, etc., kinematic graphs can be applied to linkage type synthesis [24,101], structural classification and enumeration of mechanisms [1,31], automatic mechanism design [105], and more graph theoretical approaches to kinematic chains [38]. Earl and Rooney applied kinematic graph methods to robot manipulator design. In [25], they discussed the mobility and topology of a robot manipulator from the view point of graph theory. A simplified diagram of 6-bar Watt’s linkage and its kinematic graph is shown in Fig. 3.4. Note that the ground is treated as a fixed link here.

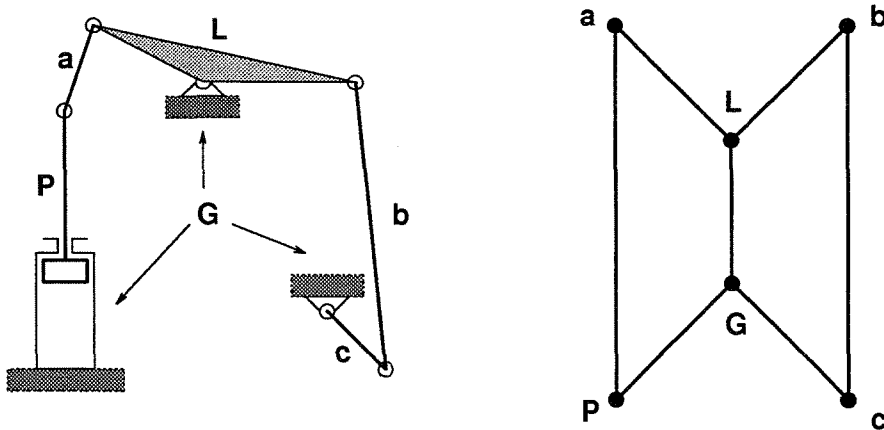


Figure 3.4: The Watt’s linkage and its kinematic graph

The regional structure of a modular robot is essentially a kinematic chain of link and joint modules. We therefore apply the same concept of kinematic graphs to represent it. This graph technique has the following advantages:

1. It provides a framework for the study of underlying geometric and kinematic properties of a robot, for example, the functional difference between a star-shaped

graph topology (a multifinger robot hand) and a serial topology (an industrial manipulator). The next chapter will show that the entire modular robot assembly can be fully described by adding more structure and features to the incidence matrix of a graph.

2. Graph theory and numerous graph algorithms can be directly applied to the enumeration and classification of modular robot assembly configuration, and kinematics of modular robots.

In modular robot applications, we are concerned with two types of robot: *homogeneous* modular robots consist of links and joints of a single type; *hybrid* modular robots consist of different types of joints and links. A labeled graph is sufficient to represent a homogeneous robot, while a specialized graph is required to represent a hybrid robot. In the specialized graph, the set of vertex types, \tilde{V} , contains the types of link modules whereas the set of edge types, \tilde{E} , contains the types of joint modules. A homogeneous robot example is shown in Fig. 3.5. A hybrid robot is shown in Fig. 3.6. Their incidence matrices are

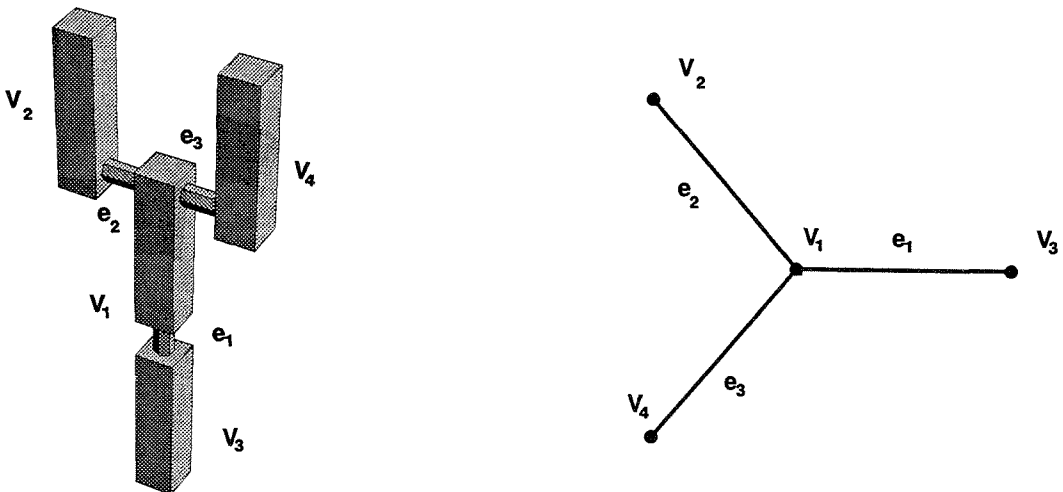


Figure 3.5: A homogeneous modular robot and its graph

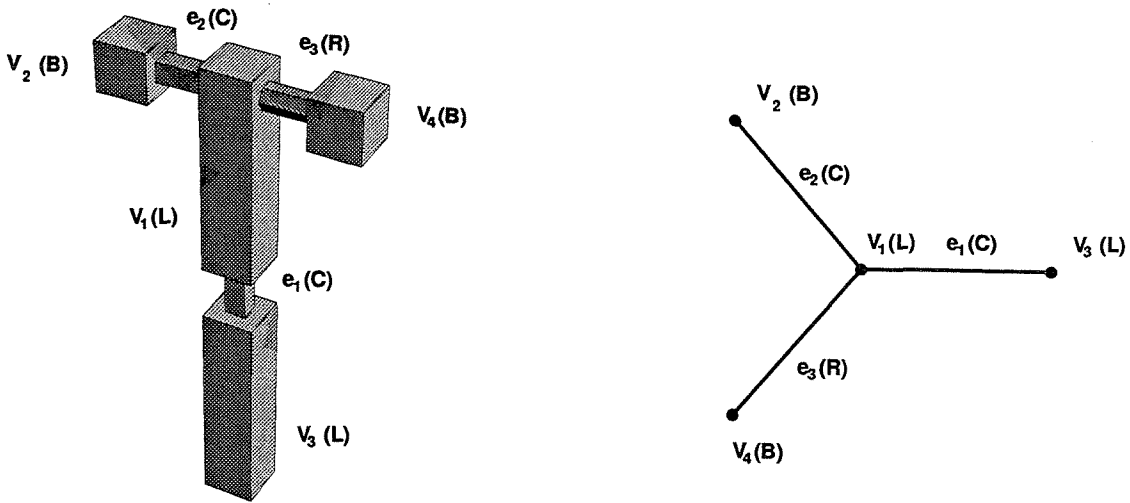


Figure 3.6: A hybrid modular robot and its graph

$$M(G) = \begin{matrix} & e_1 & e_2 & e_3 \\ v_1 & \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ v_2 & \\ v_3 & \\ v_4 & \end{matrix} \quad M(\tilde{G}) = \begin{matrix} & e_1 & e_2 & e_3 \\ v_1 & \begin{pmatrix} 1 & 1 & 1 & L \\ 0 & 1 & 0 & B \\ 1 & 0 & 0 & L \\ 0 & 0 & 1 & B \\ C & C & R & 0 \end{pmatrix} \\ v_2 & \\ v_3 & \\ v_4 & \end{matrix}, \quad (3.17)$$

3.6. Discussion

In this chapter, several basic mathematical concepts for modular robot assembly problems has been reviewed. The rotations of a symmetric link module has been related to permutations on its set of connecting ports. A method to count distinct patterns on a symmetric object was introduced as well. These methods pave the way for enumerating joint assembly patterns on a single link in the next chapter. The single link assembly is a building block for the large regional structure of a modular robot. Kinematic graphs are then applied to describe this large structure. In the next chapter we will show, by adding more features to the graph incidence matrix representation, the entire robot assembly can be fully described.

Chapter 4

Enumeration of Assembly Configurations

In this chapter, a methodology to enumerate non-isomorphic modular robot assembly configurations from a given set of modules is proposed. A formal definition of modular robot assembly configurations in terms of *assembly incidence matrices* (AIM) is introduced. This matrix representation is based on the underlying kinematic graph of the robot.

We start the enumeration procedure by finding distinct joint assembly patterns on a single link. Due to geometric symmetry, many joint patterns are alike. An equivalence relation based on the symmetric rotation group of the link is introduced to classify distinct patterns. Pólya's theorem provides a closed form formula on the number of distinct patterns. A two-stage equivalence relation built on graph isomorphisms and symmetric rotation groups of link modules is proposed to classify distinct AIMs. Distinct AIMs are in different equivalence classes with different kinematic performances. An algorithm is proposed to find all distinct assembly configurations of an n -link tree-structured modular robot. This algorithm has been implemented in *Mathematica*, and examples are given.

Section 4.1 discusses the enumeration of joint assembly patterns. An algorithm is given for listing distinct patterns. The effect of joint limits are also discussed. Section 4.2 defines the AIM and propose a two-stage equivalence relation on AIMs for classification.

Section 4.3 introduces the total algorithm for the enumeration of non-isomorphic n -link tree robot assembly configurations. Examples and comparison between brute force enumeration and our enumeration algorithm are given. Complexity issues of this algorithm are also discussed. Section 4.4 addresses issues regarding closed-loop module constructions.

4.1. Enumerating Joint Assemblies on Links

Since a link module possesses multiple connecting ports, joints can be attached to it in many different ways. The way joints are attached to the link is called an *assembly pattern*. The symmetry in link geometry and connecting port locations causes some assembly patterns to look alike after a symmetric rotation. Although the ports are marked with different numbers, the symmetry allows a link to be reoriented in such a way that these similar patterns function identically in a large robot structure.

For instance, Fig. 4.1 shows three possible ways to connect an R-joint and an H-joint to a prism link. Although the R-joint of (A) is in Port 1, that of (B) is in Port 3, and the H-joint of (A) is in Port 2; that of (B) is in Port 4, patterns (A) and (B) are actually functioning identically in a large robot assembly. Rotating (A) about its longitudinal axis by 90° matches the pattern of (B). However, (A) and (C) are entirely different because it is impossible to rotate (A) to match the pattern in (C).

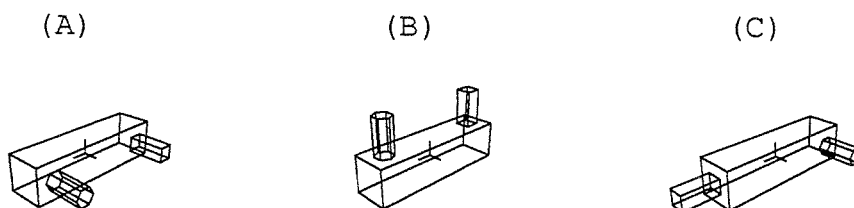


Figure 4.1: Three assembly patterns on a prism

In constructing and enumerating modular robot assemblies, we are only interested in the physically distinct patterns of link assemblies, because they will result in different robot kinematics. The link assembly problem resembles the cube coloring problem of

Section 3.3. In this case, the colors to be painted on the cube are replaced by joints attached to the connecting ports on the link. Counting distinct assembly patterns on a link can then be done by Pólya's theorem. With the number of patterns in hand, listing these patterns can be done by a computer algorithm stated in Section 4.1.2.

4.1.1. Assembly Patterns

Let PORT be the set of port numbers on a link and ATT be the set of connecting status on a port. ATT contains types of joints that are available in the modular robot system and a zero indicating an empty port. For our modular robotic system, $\text{ATT} = \{0, R, H, C, P\}$.

Definition 4.1: The *assembly pattern* on a link module is an injection mapping

$$f : \text{PORT} \rightarrow \text{ATT}. \quad (4.1)$$

The assembly pattern, f , fully describes the connecting status of the connecting ports on the link module. If a port is connected to a joint, f will assign the type of that joint to it. If the port is an empty one, a zero will be assigned. Table 4.1 shows the three assembly patterns in Fig. 4.1.

| Patterns | Ports | | | | | | | | | |
|----------|-------|---|---|---|---|---|---|---|---|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| f_a | R | H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| f_b | 0 | 0 | R | H | 0 | 0 | 0 | 0 | 0 | 0 |
| f_c | 0 | R | 0 | 0 | 0 | 0 | 0 | 0 | H | 0 |

Table 4.1: Assembly patterns in Fig. 4.1

Let $\mathcal{X} = \text{ATT}$, $\mathcal{I} = \text{PORT}$ and $n = |\text{PORT}|$, $m = |\text{ATT}|$. Since a port can be assigned with any element in ATT , for a link module with connecting ports numbered by PORT , there are m^n possible assembly patterns. The set of these assembly patterns is denoted by $\mathcal{F} \equiv \mathcal{X}^{\mathcal{I}}$. The port index set, PORT , is analogous to the set of face, and the set of connecting status, ATT , to the set of colors respectively in the cube coloring problem. A lot of patterns in \mathcal{F} are physically identical due to symmetry in the link geometry.

The following equivalence relation is defined in order to classify distinct joint assembly patterns.

Definition 4.2: Two assembly patterns $f_i, f_j : \text{PORT} \rightarrow \text{ATT}$, are *equivalent* iff there exists a symmetric rotation, $\pi \in \mathcal{S} \approx \mathcal{R}$, on the link module such that

$$f_i = f_j \circ \pi \quad (4.2)$$

Let π be a rotation of the prism about its z-axis by 90° . For assembly patterns f_a and f_b in Table 4.1, we have

$$f_a(1) = f_b \circ \pi(1) = f_b(3) = R$$

$$f_a(2) = f_b \circ \pi(2) = f_b(4) = H$$

$$f_a(k) = f_b \circ \pi(k) = 0, \quad k = 3, 4, \dots, 10$$

Hence, $f_a = f_b \circ \pi$, f_a and f_b are equivalent.

This equivalence relation divides the set of assembly patterns \mathcal{F} into disjoint subsets, the equivalence classes. Let $[f]$ denote the equivalence class containing f ; then $[f]$ represents a distinct joint assembly pattern. The set, \mathcal{F}/\mathcal{S} (or \mathcal{F}/\mathcal{R}), represents the set of distinct assembly patterns (or the set of equivalence classes under the symmetric rotation group \mathcal{R}).

\mathcal{F} contains joint assembly patterns of all types given in ATT regardless of the number of joints in each type. In most cases, one just wants to know the number of distinct assembly patterns, assuming the number of joints in each type is specified. Suppose there are k types of joints in ATT and there are d_i Type- i joints. Let $d_s = \sum_{i=1}^k d_i$, where $d_s \leq n$ and $(n - d_s)$ represents the number of empty ports on the link. We use a $(k + 1)$ -vector: $(n - d_s, d_1, d_2, \dots, d_k)$ to represent the number of joints in each type and the number of empty ports on a link module.

The number of distinct assembly patterns with a given number of joints in each type $(n - d_s, d_1, d_2, \dots, d_k)$ can be found by the following procedure:

1. Based on the features on the link, find its symmetric rotation group, \mathcal{R} , and the corresponding permutation group, \mathcal{S} .

2. Obtain the cycle index formula for \mathcal{S} from Definition 3.11.
3. Assign dummy variables y_i to every element in ATT, i.e., y_0 to 0, y_i to Type- i joint, etc.
4. Apply Pólya's theorem to find the pattern inventory, I_P . The coefficient of the term $y_0^{(n-d_s)} y_1^{d_1} y_2^{d_2} \dots y_k^{d_k}$ is the number of distinct patterns.

From combinatorics, we know there are $\frac{n!}{d_1! \dots d_k! (n-d_s)!}$ possible assembly patterns for $(n - d_s, d_1, d_2, \dots, d_k)$ joints attached to a link. This is known as the permutations of a multiset problem in [89]. Using the *backtrack* algorithm [89], one can exhaustively construct those assembly patterns without considering link symmetry. However, this number is far greater than the number of distinct ones.

Example 4.3: For the example in Fig. 4.1, the connecting status set $\text{ATT} = \{0, R, H\}$, because only R and H-joints are considered. An assembly pattern can be written as $f : \text{PORT}(L) \rightarrow \text{ATT}$. Now assign y_0 to 0, y_1 to R-joints, and y_2 to H-joints. According to Pólya's theorem, the pattern inventory is

$$\begin{aligned}
 I_P &= P_{\mathcal{S}_L}(x_1, x_2, \dots, x_n) = P_{\mathcal{S}_L}(x_1, x_2, x_4) \\
 &= P_{\mathcal{S}_L}(y_0 + y_1 + y_2, y_0^2 + y_1^2 + y_2^2, y_0^4 + y_1^4 + y_2^4) \\
 &= y_0^{10} + 2y_0^9 y_1 + 9y_0^8 y_1^2 + \dots + y_1^{10} + 2y_0^9 y_2 \\
 &\quad + 12y_0^8 y_1 y_2 + 46y_0^7 y_1^2 y_2 + 106y_0^6 y_1^3 y_2 + 160y_0^5 y_1^4 y_2 + \dots \quad (4.3)
 \end{aligned}$$

The number of distinct assembly patterns with 1 R-joint and 1 H-joint is 12, which is the coefficient of the $y_0^8 y_1 y_2$ term in (4.3). These 12 distinct patterns are shown in Fig. 4.2. A brute force enumeration of the possible assembly patterns which does not account for these symmetry effects results in $10 \cdot 9 = 90$ assembly patterns. Similarly, the coefficient of term $y_0^6 y_1^3 y_2$ indicates that there are 106 unique ways to attach 3 R-joints and 1 H-joint to a prism link. Brute force enumeration would result in $\frac{10!}{6! 3! 1!} = 840$ different, but not unique, assembly patterns. ■

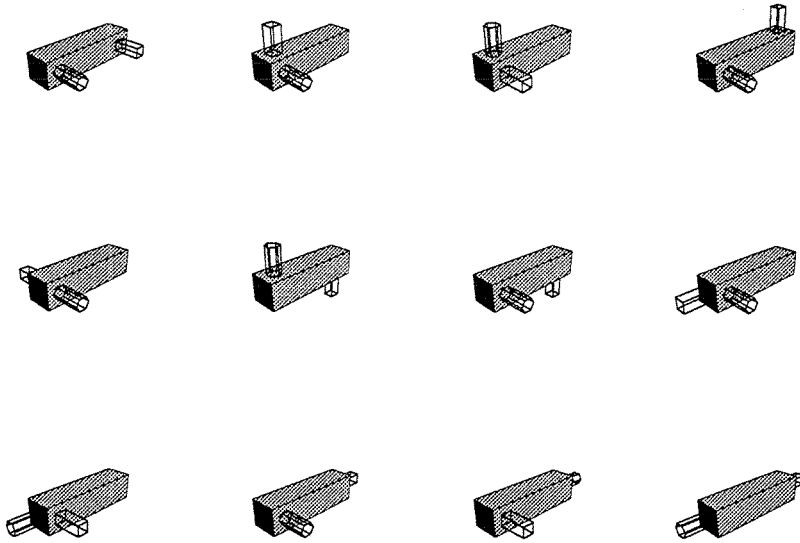


Figure 4.2: Distinct patterns for a prism with 1 R- and 1 H-joint

4.1.2. Algorithms for Listing Distinct Assembly Patterns

While the number of distinct assembly pattern can be easily obtained from Pólya's counting theorem, a listing algorithm is necessary for real applications. Distinct assembly patterns represent different equivalence classes. Since an equivalence class usually contains more than one element, listing the distinct patterns is equivalent to finding one element in each equivalence class as a representative. There has been much research on finding the minimal representative system under group actions [99, 100]. Our algorithm `OrbitEnumerate` stated below in pseudo code is a simple one which will list distinct joint assembly patterns under the action of a symmetry rotation group. `OrbitEnumerate` accepts two arguments: a set of assembly patterns, \mathcal{F} , and a symmetry rotation group, \mathcal{R} , written in permutation group form. The output of `OrbitEnumerate` is a list of distinct assembly patterns. Each pattern represents an equivalence class in \mathcal{F} . Let $\mathcal{F} = \{f_1, \dots, f_{|\mathcal{F}|}\}$ and $\mathcal{R} = \{\pi_1, \dots, \pi_m\}$, where $|\mathcal{F}|$ is the total number of assembly patterns. The algorithm works as follows.

```

0  Procedure OrbitEnumerate( $\mathcal{R}, \mathcal{F}$ )
1  Queue =  $\{1, 2, \dots, |\mathcal{F}|\}$ ;

```

```

2   NewOrbit =  $\emptyset$ ;
3   While Length(Queue) > 1 do
4       {
5       v = First(Queue);
6       Queue = Rest(Queue);
7       VOrbit =  $\emptyset$ ;
8       For all  $\pi_i \in \mathcal{R}$  do
9           { Append( $f_v \circ \pi_i$ , VOrbit) };
10      tmp = Queue;
11      For all  $i \in tmp$  do
12          { If  $f_i \in VOrbit$  then Queue = Delete( $i$ , Queue) };
13      Append( $f_v$ , NewOrbit);
14      };
15  If Queue = { $k$ } then Append( $k$ , NewOrbit);
16  Return(NewOrbit);

```

Queue is the index set of assembly patterns. NewOrbit stores distinct assembly patterns. Line 1 and 2 initialize Queue and NewOrbit. Line 5 takes out the first element, v , in Queue. f_v is the assembly pattern of v . Line 6 stores the remaining elements as Queue. Line 8-9 generate an equivalence class containing f_v , which consists of all patterns equivalent to f_v in \mathcal{F} under the action of \mathcal{R} . These equivalent patterns are collected as a set VOrbit. Line 11-12 test the remaining patterns in \mathcal{F} to determine if they are in VOrbit. If they are, then delete the indices of those patterns in Queue and update Queue. If not, then skip. After this step, the updated Queue contains no indices of patterns that are equivalent to f_v . Line 13 stores f_v in NewOrbit to represent its equivalence class.

The program then returns to Line 3 in order to extract the first element in the updated Queue and subsequently find the next distinct pattern. The loop from Line 3 to Line 14 ends when Queue has only one element left or becomes empty after removing patterns equivalent to f_v . If there is one element k left in Queue, then f_k must represent a distinct pattern, and it is stored in NewOrbit.

From experiment, we observed that OrbitEnumerate lists equivalence classes of \mathcal{F} under \mathcal{R} in $O(N_{eq}^2)$ time for a fixed \mathcal{R} , where N_{eq} is the number of equivalence classes. N_{eq} can be determined from the coefficients of the pattern inventory I_P from Theorem 3.12 when the number of joints in each type is specified.

Note that in a large robot assembly problem, the number of joints in each type on every link will be given, so the enumeration of distinct assembly patterns will be done in advance. Those distinct joint assembly patterns can be stored in a “look-up” table for the enumeration of modular robot assemblies.

4.1.3. A Note on Incorporating R-Joints with Joint Limit

In practice, a revolute joint often cannot rotate freely throughout its 360° range. The angles that cannot be accessed form a *forbidden sector* (Fig. 4.3). When such a joint is incorporated into a modular robot, the locations of the forbidden sector with respect to the link module introduce more assembly patterns as shown in Fig. 4.4.

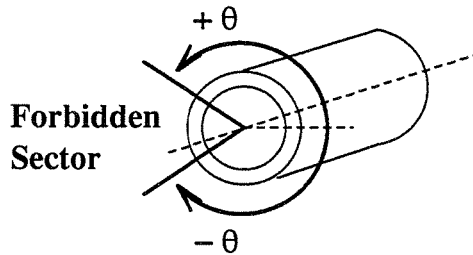


Figure 4.3: Forbidden sector of an R-joint with joint limit

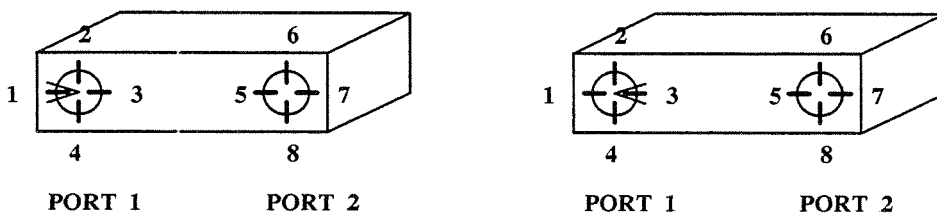


Figure 4.4: Forbidden sectors in different orientations

As mentioned in Section 3.2, by adding more features to a symmetric object, the symmetric rotation group of the object can be expressed by different permutation groups. If an R-joint with limited motion is allowed to be connected to a port in several different symmetric orientations, one can use the permutations of these orientations to represent symmetric rotations on the link module.

For instance, four symmetric orientations of the forbidden sector are defined on each of the ten ports of a prism as shown in Fig. 4.4. Totally there are 40 different orientations. The orientations on Port n are marked with $4n - 3$, $4n - 2$, $4n - 1$, and $4n$ respectively. A joint can be connected to the link in any one of these 40 orientations. A symmetric rotation can be represented by a permutation on the index set $\mathcal{I} = \{1, 2, \dots, 40\}$.

Denote the set of orientations of the forbidden sector by **ORIENT** and the R-joint with limited motion by the R_l -joint. When only R_l -joints are considered, the assembly pattern can be expressed as an injection mapping, $f : \mathbf{ORIENT} \rightarrow \{0, R_l\}$. A symmetric rotation can be expressed as a permutation, $\pi : \mathbf{ORIENT} \rightarrow \mathbf{ORIENT}$. The definition of an equivalence relation on assembly patterns under the action of this new permutation group is similar to Definition 4.2. For example, the assembly patterns in Fig. 4.4 are shown in Table 4.2.

| Patterns | Orientations | | | | | | |
|----------|--------------|---|-------|---|---|-----|----|
| | 1 | 2 | 3 | 4 | 5 | ... | 40 |
| f_1 | R_l | 0 | 0 | 0 | 0 | ... | 0 |
| f_2 | 0 | 0 | R_l | 0 | 0 | ... | 0 |

Table 4.2: Assembly patterns for R_l -joints in Fig. 4.4

However, in the non-homogeneous cases where joints with and without joint limits are considered, for example, R-joints and R_l -joints, the assembly pattern must be defined in a different fashion. In the case of an R-joint, all orientations on the connected port should be marked occupied since an R-joint can freely access these orientations and it is impossible to put another R_l -joint in any of these orientations. These occupied orientations represent only one R-joint whereas one orientation is needed for an R_l -joint. The same situation applies when other joints not using this feature are employed, such as C and H-joints. Therefore, it is possible that a mapping from **ORIENT** to a combined joint type set is not a physically realizable assembly pattern.

Follow the above example with four different orientations on a port. If we put an

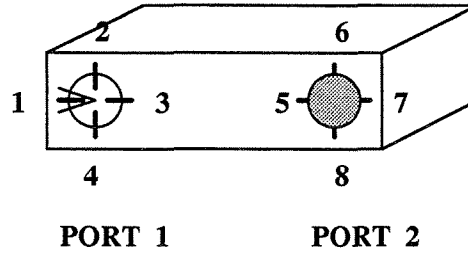


Figure 4.5: Assembly pattern with an R-joint and an R_l -joint

R-joint in Port 2 and an R_l -joint in Port 1 as shown in Fig. 4.5, the assembly pattern will be

$$f : \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \dots & 40 \\ R_l & 0 & 0 & 0 & R & R & R & R & 0 & 0 & \dots & 0 \end{pmatrix} \quad (4.4)$$

Note that although Orientations 5, 6, 7, and 8 are all associated with R, they actually represent one R-joint on Port 2. The following mapping from ORIENT to $\{0, R_l, R\}$ is not an assembly pattern because Orientation 6 is empty and no physically realizable joint patterns can match it.

$$f : \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \dots & 40 \\ R_l & 0 & 0 & 0 & R & 0 & R & R & 0 & 0 & \dots & 0 \end{pmatrix} \quad (4.5)$$

Let ATT be the set of joint types including R_l -joint. Denote $\mathcal{X} \equiv \text{ATT}$ and $\mathcal{I} \equiv \text{ORIENT}$. From the above discussion, we conclude that only a subset of $\mathcal{X}^{\mathcal{I}}$ is the set of assembly patterns with the extended feature ORIENT. Thus, Pólya's theorem cannot be applied in this case to find the number of distinct assembly patterns. However, the algorithm OrbitEnumerate can still be applied for listing the distinct patterns.

4.2. Assembly Incidence Matrices

The topology of a modular robot can be described by a labeled graph or a specialized graph which, in turn, can be expressed by an incidence matrix (IM) or an extended incidence matrix (eIM). The entire robot assembly problem concerns not only the topology but also the joint assembly patterns on every link. Hence, by substituting the connected port numbers into the non-zero entries of the incidence matrix, the joint patterns on all link modules can be fully described. The resulting matrix is similar to an

incidence matrix in structure but contains joint assembly patterns as well. This matrix fully describes a robot assembly configuration, and is termed an *assembly incidence matrix*.

Definition 4.4: Let G be the graph of a homogeneous modular robot. The assembly configuration of this robot is defined to be an assembly incidence matrix, $A(G)$, obtained by replacing every entry of 1 in $M(G)$ with a non-zero integer, $k \in \text{PORT}$. The zero entries remain unchanged. $a_{ij} = k$ indicates that joint e_j is attached to port k of link v_i ; $a_{ij} = 0$, otherwise.

Similar to homogeneous robot case, the assembly configuration of a hybrid robot can be defined as an eIM-like matrix with information on connected ports. We call this matrix an *extended assembly incidence matrix* (eAIM).

Definition 4.5: Let \tilde{G} be the specialized graph of a hybrid robot. The assembly configuration of this robot is defined to be an extended assembly incidence matrix, $A(\tilde{G})$, obtained by replacing every entry of 1 in $M(\tilde{G})$ with a non-zero integer, $k \in \text{PORT}$. The zero entries remain unchanged. $a_{ij} = k$ indicates that joint e_j is attached to port k of link v_i . Note that a_{ij} must belong to the port index set on link v_i because the type of links in a hybrid robot may be different. $a_{ij} = 0$, otherwise.

Example 4.6: The AIMs of the modular robots shown in Fig. 3.5 and 3.6 are

$$A(G) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{ccc} e_1 & e_2 & e_3 \\ \left(\begin{array}{ccc} 10 & 5 & 1 \\ 0 & 2 & 0 \\ 9 & 0 & 0 \\ 0 & 0 & 4 \end{array} \right) \end{array} \quad A(\tilde{G}) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{ccc} e_1 & e_2 & e_3 \\ \left(\begin{array}{ccc} 10 & 5 & 1 \\ 0 & 1 & 0 \\ 9 & 0 & 0 \\ 0 & 0 & 2 \\ C & C & R \end{array} \right) \end{array} \cdot \blacksquare \quad (4.6)$$

Note that row vectors in an AIM are closely related to the joint assembly patterns defined in Section 4.1.1. The i^{th} row vector shows connection information of all joints on link v_i . If joint e_j is connected to this link, the j^{th} entry tells which port it is connected to. Therefore, the joint assembly pattern on link v_i is implied.

Remark: When prismatic joints are incorporated in a modular robot system, the relative orientation of two links connected by a P-joint is always stationary during the translational motion. The number of connected port information in an AIM is not sufficient to express this orientation. The concept of using extended features on a link module for R_f -joints can be applied here. By marking a P-joint and allowing it to be attached to the connecting port in several symmetrical orientations, we are able to describe a finite number of the relative link orientations. For example, if a P-joint is allowed to be attached to a port in four different orientations as shown in Fig. 4.4, there will be four different connection methods for the two links shown in Fig. 4.6. The extended feature can be stored in the AIM as part of the assembly configuration.

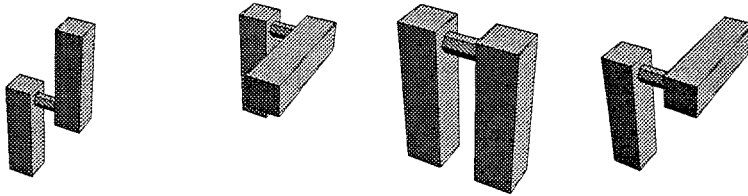


Figure 4.6: Relative orientations of two prisms with a P-joint

4.2.1. Equivalence of AIMs

The AIM is an algebraic representation of a modular robot assembly configuration. When the physical construction of a modular robot is considered, isomorphisms of the underlying graph and link symmetries will make different AIMs having identical kinematic properties, such as the size and shape of the workspace and joint singularities. The graph isomorphism induces permutations on columns and rows of AIMs which will alter the locations of non-zero entries. Link symmetry allows link modules to be reoriented in such a way that the physical connection pattern of the joints remain unchanged. We show here an example of three different AIMs which lead to physically identical robot construction.

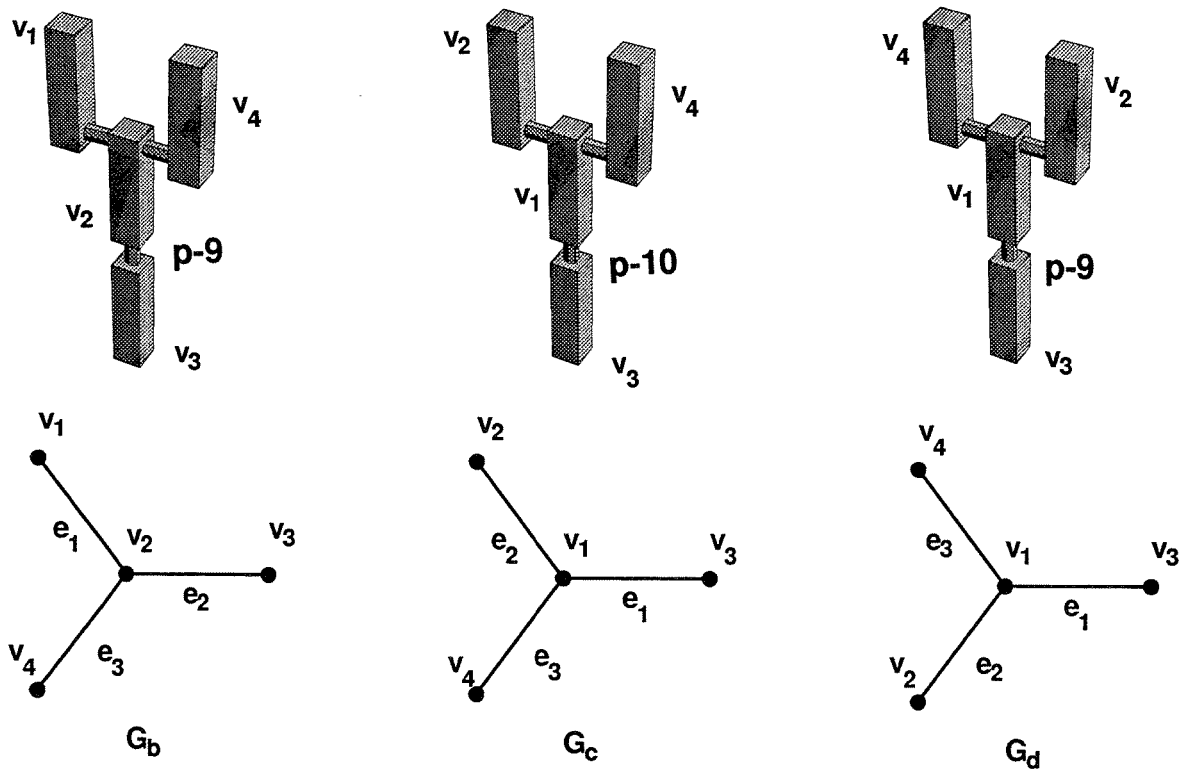


Figure 4.7: Three physically identical configurations

Example 4.7: The three robots shown in Fig. 4.7 are constructed from the three different AIMs. Removing all the labels and numbers of the connected ports, they are identical to the robot shown in Fig. 3.5.

$$A(G_b) = \begin{matrix} & e_1 & e_2 & e_3 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} & \begin{pmatrix} 2 & 0 & 0 \\ 5 & 10 & 1 \\ 0 & 9 & 0 \\ 0 & 0 & 4 \end{pmatrix} & & & \\ \end{matrix} \quad
 A(G_c) = \begin{matrix} & e_1 & e_2 & e_3 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} & \begin{pmatrix} 10 & 5 & 1 \\ 0 & 2 & 0 \\ 10 & 0 & 0 \\ 0 & 0 & 4 \end{pmatrix} & & & \\ \end{matrix} \quad
 A(G_d) = \begin{matrix} & e_1 & e_2 & e_3 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} & \begin{pmatrix} 10 & 1 & 5 \\ 0 & 4 & 0 \\ 9 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} & & & \\ \end{matrix} \quad (4.7)$$

Denote the graph in Fig. 3.5 by G_a . From Definition 3.16, G_a , G_b , G_c , and G_d are isomorphic graphs. We can see that $A(G_a)$ and $A(G_b)$ are different by the permutation $\gamma_{ba} = ((v_2, v_1, v_3, v_4), (e_2, e_1, e_3))$. All entries of $A(G_c)$ and $A(G_a)$ are equal except entry (3,1). This entry indicates the number of the connected port on Link v_3 . Since this link is a symmetric square prism, using either Port 9 or Port 10 represents a physically identical pattern. ■

We now define an equivalence relation on AIMs for the purpose of classification. Because an AIM describes both topology and joint assembly pattern information, the equivalence relation will be established in two stages: robot topology equivalence and pattern equivalence on corresponding link modules. Topology equivalence insures that the graphs of robot assemblies are isomorphic. AIMs with non-isomorphic graphs are definitely distinct. Pattern equivalence will be checked on every link module. AIMs with topology equivalence will look identical only when all joint patterns are matched on corresponding links.

We state the equivalence on AIMs of homogeneous modular robot in detail. The equivalence on eAIMs of hybrid robots follows.

Homogeneous Robots

Let G_1 and G_2 be the graphs of two robots, and $A(G_1)$ and $A(G_2)$ their respective AIMs.

1. Topology Equivalence

Definition 4.8: Two AIMs $A(G_i)$, $i = 1, 2$, are *topologically equivalent*, iff G_1 and G_2 are isomorphic.

Graph isomorphism induces permutations on columns and rows of AIMs. These permutations represent the relabeling process on links and joints which does not alter the physical connections of links and joints. Therefore, the number of connected ports remain unchanged during permutation action.

2. Pattern Equivalence

Suppose $A(G_1)$ and $A(G_2)$ are topologically equivalent and γ_{12} is the isomorphism from G_1 to G_2 . After the permutations of columns and rows according to γ_{12} , the locations of non-zero entries in both $A_{\gamma_{12}}(G_1)$ and $A(G_2)$ are identical. Then we are able to compare the assembly patterns on the links. Denote \hat{w}_i^1 and \hat{w}_i^2 the i^{th} row vector of $A_{\gamma_{12}}(G_1)$ and $A(G_2)$ respectively.

Definition 4.9: Two row vectors $\widehat{w}_i^1 = (a_{i1}^1, \dots, a_{in}^1)$, and $\widehat{w}_i^2 = (a_{i1}^2, \dots, a_{in}^2)$, are *pattern equivalent*, iff there is a symmetric rotation (written in a permutation of port index set), $\pi : \text{PORT} \rightarrow \text{PORT}$, on Link v_i such that

1. for non-zero entry, $a_{ij}^1 \in \text{PORT}$, $\pi(a_{ij}^1) = a_{ij}^2$;
2. for zero entry, $a_{ij}^1 = a_{ij}^2 = 0$.

If these two rows are equivalent, they represent physically identical joint assembly patterns on link v_i . In Example 4.7, rows 1,2, and 4 of $A(G_a)$ and $A(G_c)$ are the same so they are equivalent. For $A(G_c)$, rotating prism v_3 about its x-axis by 180° transforms the 3rd row, $(10, 0, 0)$, into $(9, 0, 0)$, which is identical to the 3rd row of $A(G_a)$, so $(10, 0, 0)$ and $(9, 0, 0)$ are pattern equivalent.

3. A Note on Graph Automorphisms

When a robot has topological symmetry, i.e., its kinematic graph has symmetry and possesses automorphisms, the automorphism will induce a row and column permutation which renders its AIM similar to itself. That means the locations of non-zero entries will not be altered after an automorphism action. Hence, one must check all automorphisms when comparing for the pattern equivalence on rows of topologically equivalent AIMs.

Let $A(G_1)$ and $A(G_2)$ be topologically equivalent AIMs and γ_{12} is the isomorphism from G_1 to G_2 . If there exists an $\eta \in \mathcal{H}(G_2)$ which makes all corresponding rows of $A_{\gamma_{12}}(G_1)$ and $A_\eta(G_2)$ equivalent in pattern, then $A(G_1)$ and $A(G_2)$ are equivalent.

Definition 4.10: (*Equivalence of AIM*)

Two AIMs $A(G_1)$ and $A(G_2)$ are equivalent if and only if they are topologically equivalent, i.e., G_1 and G_2 are isomorphic, and there exist an automorphism of G_2 , $\eta \in \mathcal{H}(G_2)$, such that all corresponding row vectors in $A_{\gamma_{12}}(G_1)$ and $A_\eta(G_2)$ are equivalent in pattern, where γ_{12} is an isomorphism from G_1 to G_2 . We write it as

$$A(G_1) \sim A(G_2). \quad (4.8)$$

Note that η may be the identity automorphism if the underlying graph G has no symmetry.

In Example 4.7, the automorphism $\eta = ((v_1, v_4, v_3, v_2), (e_1, e_3, e_2))$ of G_d makes AIMs $A_\eta(G_d) = A(G_a)$, so $A(G_a)$ and $A(G_d)$ are equivalent.

Hybrid Robots

The equivalence of eAIMs of hybrid robots are established in a similar way. Let \tilde{G}_1 and \tilde{G}_2 be the specialized kinematic graphs of two hybrid robots, and $A(\tilde{G}_1)$ and $A(\tilde{G}_2)$ be the eAIMs.

1. Topology Equivalence

Definition 4.11: Two eAIMs $A(\tilde{G}_i)$, $i = 1, 2$, are *topologically equivalent*, iff \tilde{G}_1 and \tilde{G}_2 are isomorphic.

2. Pattern Equivalence

Let \hat{w}_i^1 and \hat{w}_i^2 be the i^{th} row vectors of the upper-left $m \times n$ submatrix of $A_{\gamma_{12}}(\tilde{G}_1)$ and $A(\tilde{G}_2)$ respectively. The definition of pattern equivalence on \hat{w}_i^1 and \hat{w}_i^2 is similar to Definition 4.9 only that the symmetric rotation action is determined according to the type of link v_i .

3. A Note on Graph Automorphisms

A specialized graph may have automorphisms if its topology is symmetric. Likewise, when checking for pattern equivalence on rows of topologically equivalent eAIMs, one has to consider all automorphisms of \tilde{G}_1 and \tilde{G}_2 .

Definition 4.12: (*Equivalence of eAIM*)

Two eAIMs $A(\tilde{G}_1)$ and $A(\tilde{G}_2)$ are equivalent iff they are topologically equivalent, and there exists an automorphism of \tilde{G}_2 , $\eta \in \mathcal{H}(\tilde{G}_2)$, such that all corresponding rows in $A_{\gamma_{12}}(\tilde{G}_1)$ and $A_\eta(\tilde{G}_2)$ are equivalent in pattern, where γ_{12} is the isomorphism from \tilde{G}_1 to \tilde{G}_2 .

Similar AIMs (or eAIMs) are in the same equivalence class; distinct ones are in different equivalence classes. Let $[A(G)]$ (or $[A(\tilde{G})]$) denote the equivalence class containing the assembly configuration $A(G)$ (or $A(\tilde{G})$), then $[A(G)]$ (or $[A(\tilde{G})]$) represents a distinct robot assembly configuration. Equivalent AIMs look alike and have identical robot kinematics, but the converse is not true. It will be shown in the next chapter that some inequivalent AIMs may have identical kinematics when kinematic models of modular robots are introduced.

4.2.2. Hashed Assembly Incidence Matrix

This section discusses an alternative expression of the assembly configurations of a hybrid robot. The eAIM of a robot with m link and n joints requires an $(m+1) \times (n+1)$ matrix to store topology and module type assignments. Another way to express this assembly configuration is to combine the assignments on links and joints and the connected ports into non-zero integers, replace the entries with port number in the eAIM by those integers, and remove the last row and column of the eAIM. The result is a compact $m \times n$ IM-like matrix with integer data entries easy for computer processing.

The coding scheme employs a hash function which can transform the *keys*, a_{ij} (port number on link v_i), $a_{i,(n+1)}$ (type of link v_i), and $a_{(m+1),j}$ (type of joint e_j), into a positive integer termed an *assembly hash function*. This matrix, using a hashing function to represent a hybrid robot assembly configuration, is termed a *hashed assembly incidence matrix*.

Prior to using a hashing function, we have to transform the keys—the types of links and joints, into integers so that mathematical operations can perform. Let \tilde{V} (\tilde{E}) be the set of link (joint) types and N_v (N_e), the total number of elements in \tilde{V} (\tilde{E}). We assign a unique integer from 0 to $N_v - 1$ ($N_e - 1$) to every element of \tilde{V} (\tilde{E}). Thereafter, we call the types of modules by integers instead of alphabets. For example, Type-0 link, Type-2 joint, etc.

Definition 4.13: Let $p(l)$ be the number on the connected port of link v_i where joint e_k is attached. l represents the type of link v_i , and j , the type of joint e_k . Note that $p(l) \neq 0$ and $0 \leq l \leq N_v - 1$ and $0 \leq j \leq N_e - 1$. An *assembly hash function*, $h(l, j, p(l))$, which maps the connection of link v_i and joint e_j , is defined as

$$h(l, j, p(l)) = (p(l) \cdot N_e + j) \cdot N_v + l \quad (4.9)$$

$h(l, j, p(l)) \neq 0$ because $N_l, N_j \neq 0$. The following proposition shows that h is uniquely determined by module types, l and j , and the connecting port $p(l)$. This uniqueness provides an alternative way to represent the eAIM without additional row and column to represent the link and joint module assignments.

Proposition 4.14: The assembly hash function, $h(l, j, p(l))$, is unique.

Proof: Suppose there are two connections: $[l_a, j_a, p_a(l_a)]$ and $[l_b, j_b, p_b(l_b)]$.

Let $h(l_a, j_a, p_a(l_a)) = h(l_b, j_b, p_b(l_b))$, where $0 \leq l_a, l_b \leq N_v - 1$ and $0 \leq j_a, j_b \leq N_e - 1$.

By definition, we have

$$(p_a(l_a) \cdot N_e + j_a) \cdot N_v + l_a = (p_b(l_b) \cdot N_e + j_b) \cdot N_v + l_b. \quad (4.10)$$

After rearrangement, we obtain

$$[[p_a(l_a) - p_b(l_b)] \cdot N_e + (j_a - j_b)] = \frac{l_b - l_a}{N_v}. \quad (4.11)$$

Because $0 \leq \frac{l_b - l_a}{N_v} \leq \frac{N_v - 1}{N_v} < 1$ and the left hand side of (4.11) is an integer, the only way to satisfy the equality condition is to let $l_b - l_a = 0$. Therefore, $l_a = l_b$.

Rearrange the left hand side of (4.11), and we have

$$p_a(l_a) - p_b(l_b) = \frac{j_b - j_a}{N_e}. \quad (4.12)$$

Similarly, $0 \leq \frac{j_b - j_a}{N_e} < 1$ and $p_a(l_a) - p_b(l_b)$ is an integer, so $j_a = j_b$ and $p_a(l_a) = p_b(l_b)$.

Conversely, if $l_a = l_b$, $j_a = j_b$, and $p_a(l_a) = p_b(l_b)$, then $h(l_a, j_a, p_a(l_a)) = h(l_b, j_b, p_b(l_b))$.

Therefore, we prove the uniqueness of the assembly hash function h . ■

Definition 4.15: An $m \times n$ hashed AIM $A^*(\tilde{G})$ is derived from an $(m + 1) \times (n + 1)$ eAIM $A(\tilde{G})$ using an assembly hashing function such that

1. If $a_{ij} = 0$, then $a_{ij}^* = 0$.
2. If $a_{ij} \neq 0$, then $a_{ij}^* = h(a_{i,n+1}, a_{m+1,j}, a_{ij})$, $i = 1, \dots, m$ and $j = 1, \dots, n$.

Because of the uniqueness of $h(l, j, p(l))$, one can reconstruct an eAIM from a given hashed AIM. There is a 1-to-1 correspondence between eAIMs and hashed AIMs.

Example 4.16: Suppose we have module type sets: $\tilde{V} = \{L, B\}$ and $\tilde{E} = \{R, C, P, H\}$.

Assign them with integers as follows.

$$\begin{pmatrix} L & B \\ 0 & 1 \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} R & C & P & H \\ 0 & 1 & 2 & 3 \end{pmatrix} \quad (4.13)$$

Applying Definition 4.15 to $A(\tilde{G})$ in (4.6), we obtain a hashed AIM

$$A^*(\tilde{G}) = \begin{matrix} & e_1 & e_2 & e_3 \\ v_1 & \begin{pmatrix} 42 & 22 & 4 \\ 0 & 7 & 0 \\ 38 & 0 & 0 \\ 0 & 0 & 9 \end{pmatrix} \\ v_2 & \\ v_3 & \\ v_4 & \end{matrix} \quad (4.14)$$

4.3. Enumerating Robot Assembly Configurations

In this section, an algorithm to enumerate all of the distinct n -link tree-structured modular robot configurations from a given set of modules is proposed. An n -link tree robot has $n - 1$ joints and contains no closed-loop structures.

We consider only tree-like topologies, and not topologies with closed loops, for the following reason. In general, a closed loop kinematic chain must have at least seven internal degrees of freedom in order to have finite mobility [62]. A closed kinematic chain with fewer than seven degrees of freedom will have finite mobility only if the links have a special, or nongeneric, geometry. In fact, a kinematic chain with fewer than seven DOF built from a given set of modules may not even be able to close on itself. Thus, if we generate a kinematic graph with a closed loop as a subgraph, there is no guarantee that the physical structure corresponding to the subgraph can

even be constructed, or that it will be mobile. The procedure outlined below could be extended to systems with loops if an additional procedure which checks for loop closure and mobility is added. If a generated graph has a sub-loop which is physically constructible and mobile, then it is accepted, or else it is rejected.

The tree-like robots are divided into two classes: *free flying* and *fixed base*. A free-flying robot does not have an identified base link, while a fixed base robot does. The robot's base can be considered as a different link type. Fixed base robots are thus treated as hybrid robots, with the base link location in the kinematic chain determined during the hybrid robot specialization process. Homogeneous robots are necessarily free-flying robots. Conversely, a free-flying robot may be either a homogeneous or hybrid robot.

4.3.1. The Enumeration Algorithm

Analogous to the enumeration of joint assembly patterns on a single link, the enumeration of distinct modular robot assemblies is equivalent to counting the number of equivalence classes in the set of assembly configurations generated from a set of link and joint modules. But this equivalence relation requires graph isomorphisms and symmetric rotation groups on link modules so that it does not offer a single and closed form formula for the number of configurations, as in the application of Pólya's theorem to joint assembly pattern enumeration.

Based on the equivalence relations of Definitions 4.10 and 4.12, the robot enumeration procedure is divided in two parts in order to reduce the complexity of the entire algorithm: robot topology enumeration and pattern enumeration on individual link modules.

The procedure begins with a link set, LINK, of n elements and a joint set, JOINT, of $n - 1$ elements. First, candidate robot topologies are enumerated. The algorithm will find all non-isomorphic trees (graphs) with a given number of vertices. For homogeneous robots, finding non-isomorphic labeled graphs will be enough, but for hybrid robots, one more step is required—finding distinct link and joint assignments for the specialized

graphs. In this step, the automorphism groups of the underlying graphs are required.

After all non-isomorphic (specialized) graphs are found, pattern enumeration is performed on each of these graphs. Distinct assembly patterns on every link can be found first by the `OrbitEnumerate` algorithm. From a combination of distinct assembly patterns on every link in the robot, all possible robot assembly configurations can be generated. An automorphism check will be performed among these AIMs to ensure they are inequivalent. The details of this procedure follows. The flow chart of this algorithm is shown in fig. 4.8.

Step 1: *Generate non-isomorphic trees $\{G_i\}$ and write them as $n \times m$ incidence matrices $\{M(G_i)\}$.*

Several computer algorithms have been proposed to generate non-isomorphic rooted and free trees for a given number of vertices [7,23,52,86,102]. A rooted tree corresponds to a fixed base robot with the root vertex representing the fixed base. A free tree has no root and corresponds to a free-flying robot. Beyer and Hedetniemi [7] introduced a constant time algorithm to generate all rooted trees of a given size (the number of vertices). Based on this work, Wright et al. [102] propose a constant time algorithm to generate all free trees of a given size. We need only free trees in this step.

Step 2: *Find the automorphism group, $\mathcal{H}(G_i)$, using the backtrack algorithm [89].*

Step 3: *If the module sets contain different types of modules, find distinct assignments from LINK and JOINT to the vertices and edges of G_i . From those distinct assignments, construct non-isomorphic specialized trees $\{\tilde{G}_{ik}\}$ based on G_i and represent them in eIMs, $M(\tilde{G}_{ik})$. If not, go to Step 5.*

An assignment, $f_v : V \rightarrow \text{LINK}$ and $f_e : E \rightarrow \text{JOINT}$, on a graph G_i is similar to an assembly pattern, $f : \text{PORT} \rightarrow \text{ATT}$, on a link module in form. The vertices and edges correspond to port numbers; the LINK and JOINT module type sets correspond to ATT. The automorphism group $\mathcal{H}(G_i)$ is a permutation group corresponding to the symmetric rotation group on the link. Hence, we employ `OrbitEnumerate` to find

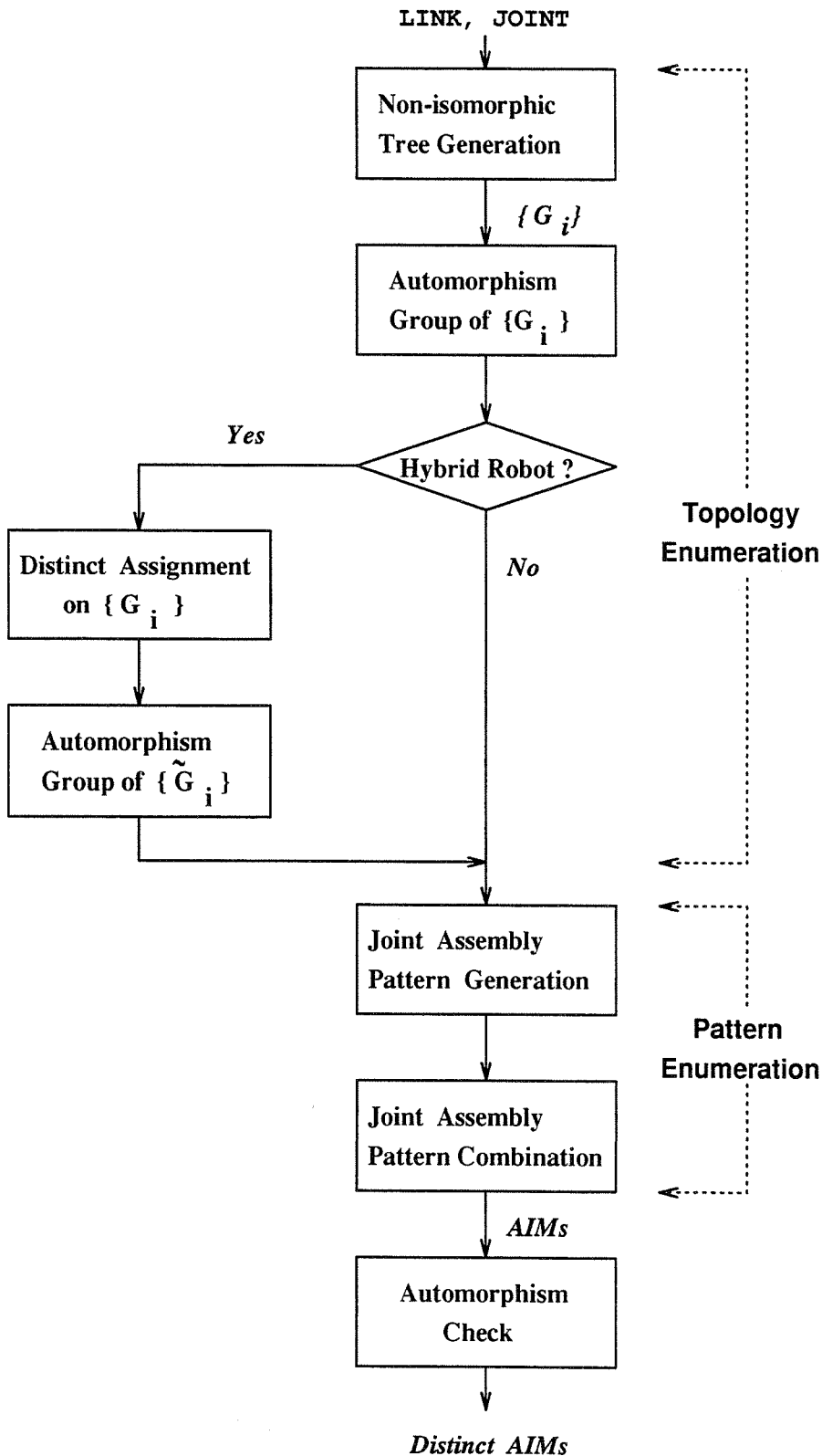


Figure 4.8: The flow chart of RobotEnumerate

distinct module assignments on G_i . The input permutation group is $\mathcal{H}(G_i)$ and the input pattern set becomes the set of vertex and edge assignments from LINK and JOINT. The graph G_i along with an assignment (f_v, f_e) becomes a specialized graph. Another way to find a distinct assignment is based on the *chain group* proposed by Yan and Hwang [105]. In their work, a heuristic algorithm based on the *chain group* of a kinematic chain is employed to enumerate non-isomorphic specialized mechanism. This chain group is similar to the automorphism group of a labeled graph.

A fixed base robot can be obtained by putting a base link in the LINK set. The location of this base link in the kinematic graph is determined by this specialization procedure.

Step 4: For every non-isomorphic specialized tree \tilde{G}_i , find $\mathcal{H}(\tilde{G}_i)$ from $\mathcal{H}(G_i)$, the automorphism group of its underlying graph G_i .

Since $\mathcal{H}(\tilde{G}_i) \subset \mathcal{H}(G_i)$, we select all automorphisms $\eta \in \mathcal{H}(G_i)$ such that $M_\eta(\tilde{G}_i) = M(\tilde{G}_i)$ to form $\mathcal{H}(\tilde{G}_i)$.

Step 5: For every non-isomorphic graph G_i (or \tilde{G}_i),

- a. Generate distinct assembly patterns for every link, based on the row vectors of $M(G_i)$ (or $M(\tilde{G}_i)$).

The number of joints in each type on a link is indicated in the row vectors of $M(G_i)$ (or $M(\tilde{G}_i)$); hence, `OrbitEnumerate` is employed to generate distinct assembly patterns on every link. All labeled joints are treated as different types of joints here. For example, the first row of $M(\tilde{G})$ of Fig. 3.6 is $(1, 1, 1, L)$, which indicates that C-joints e_1 and e_2 and R-joint e_3 are attached to prism link v_1 . Assuming these three labeled joints are different, we obtain 46 distinct joint assembly patterns on a prism. Replacing 1's in $(1, 1, 1, L)$ with port numbers from the distinct patterns, we get 46 inequivalent 1st row vector representations for an eAIM. Repeat this process for every row of $M(G_i)$.

- b. Combine all pattern inequivalent row vectors for every row of $M(G_i)$ (or $M(\tilde{G}_i)$) to construct inequivalent AIMs $A(G_i)$ (or eAIMs, $A(\tilde{G}_i)$).

- c. If the graph has no symmetry, then go to the next step. Otherwise, use $\mathcal{H}(G_i)$ (or $\mathcal{H}(\tilde{G}_i)$) to eliminate equivalent AIMs (or eAIMs) due to graph symmetry.

Step 6: Repeat Step 5 for all trees.

Step 7: Stop.

Remark 1: Steps 1 to 4 focus on topology enumeration. Steps 3 and 4 are for specialized graphs only. Steps 5 and 6 are pattern enumeration part. In chapter 6, we will show that in order to find all candidate assembly configurations with a given set of link and joint modules and robot topology for task-optimal configuration problem, we can skip steps 1 and 2 and start directly from the specialization procedure in steps 3 and 4.

Remark 2: This procedure can be transformed into a task-oriented enumeration scheme by putting constraints on generating distinct joint assemblies on every link in step 5-(a) thus reducing the number of subsequent applicable robot assembly configurations. For example, the non-redundant joint requirement eliminates joint assembly patterns where two joint axes on a link are collinear. The constraint that two joints are not allowed to be simultaneously attached to one end of the prism results in only seven distinct two-joint assemblies on a prism.

Remark 3: Brute force enumeration procedure follows the same topology enumeration part (step 1–4) stated above. The difference is in the pattern enumeration part. The symmetries on link geometry are not taken into consideration when finding assembly patterns on individual links by the brute force approach.

4.3.2. Examples

Some examples will illustrate the advantages of using this method over the straightforward approach of brute force enumeration of all possible modular robot assembly configurations.

Example 4.17: Suppose we construct a 3-link hybrid robot from module sets: LINK = $\{B, L, L\}$ and JOINT = $\{R, R\}$. The first step is to construct its non-isomorphic kinematic graphs. In this case, there is only one possible tree structure—a serially connected tree. From step 3, we find two non-isomorphic module assignments. Their specialized trees are shown in Fig. 4.9 and denoted by G_1 and G_2 .

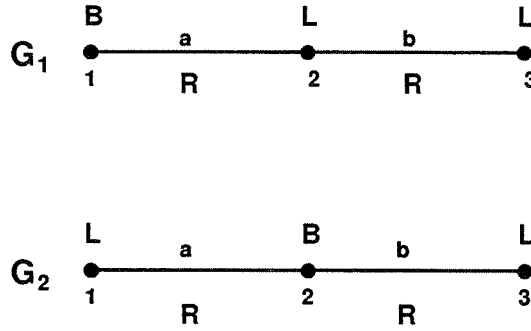


Figure 4.9: Two non-isomorphic specialized graphs

Now go to step 4 and take G_1 for example. Link 1 is a cube with one R-joint; there is only one distinct assembly pattern under this condition. There are 12 distinct assembly patterns on link 2, for a prism with two labeled joints. There are two assembly patterns on link 3 with one R-joint. Altogether at most $N_{G_1} = 1 \times 12 \times 2 = 24$ configurations can be found from G_1 . Similar for G_2 , there are at most $N_{G_2} = 2 \times 2 \times 2 = 8$ possible configurations. However, G_2 is symmetric about the center vertex. $\mathcal{H}(G_2)$ contains two nontrivial elements: the identity, $((1, 2, 3), (a, b))$, and $((3, 2, 1), (b, a))$. Using this automorphism, we further reduce the number of distinct configurations of G_2 to 6. From two non-isomorphic graphs G_1 and G_2 , there are totally $24 + 6 = 30$ distinct assembly configurations for a 3-link hybrid tree robot which are shown in Fig. 4.10.

If we do not pay attention to equivalent patterns and enumerate them in a brute force fashion, for G_1 , there are 6 patterns on link 1, $\frac{10!}{2!8!} = 45$ on link 2, and 10 on link 3. Altogether there will be $6 \times 45 \times 10 = 1800$ configurations. For G_2 , there are 10 patterns on link 1, 15 on link 2, and 10 on link 3. In total, there will be $10 \times 15 \times 10 = 1500$ configurations. There are 3300 configurations altogether. Thus, our method provides a significant improvement over brute force enumeration. ■

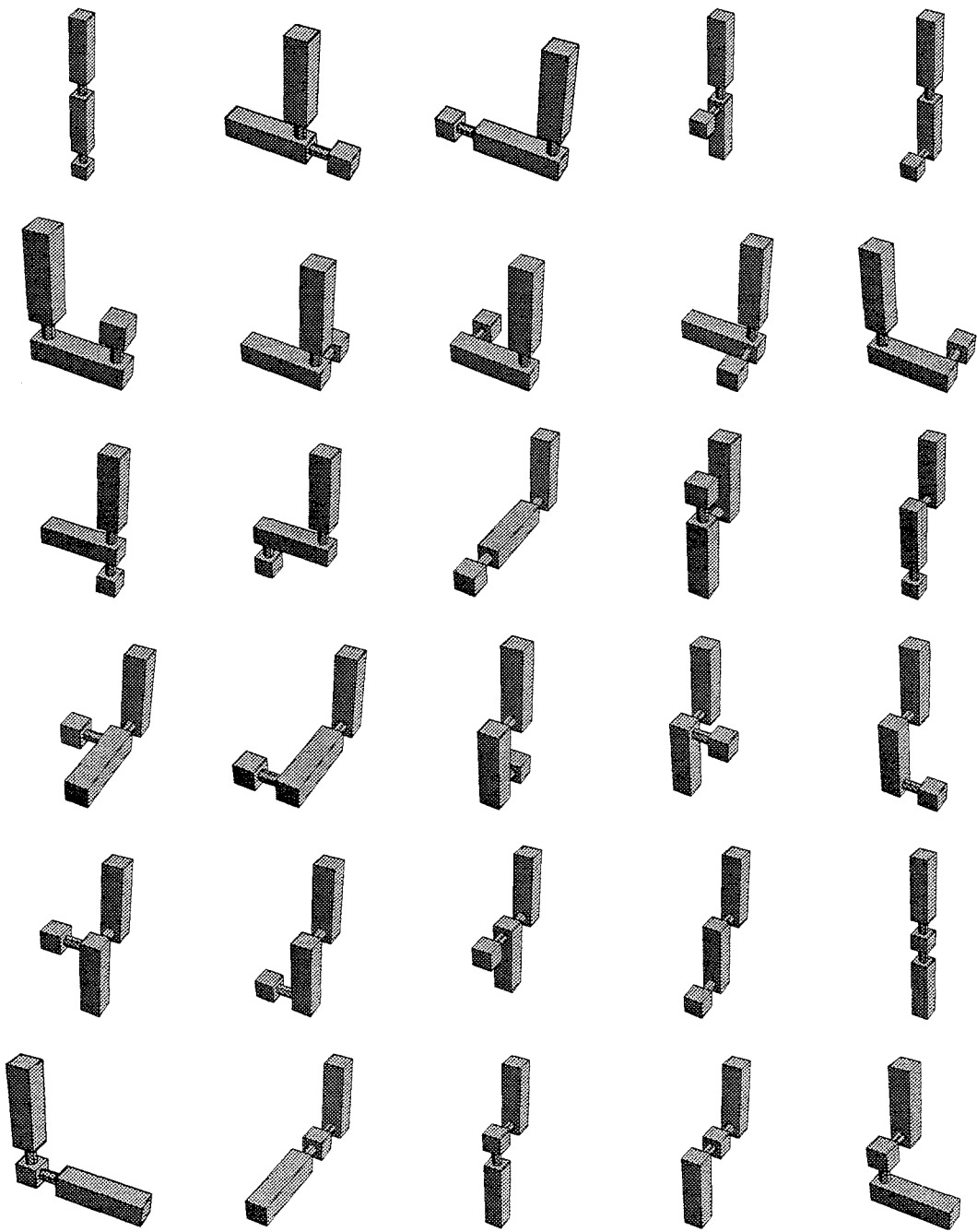


Figure 4.10: Distinct configurations of a 3-link 2-DOF hybrid robot

Example 4.18: Suppose we want to construct a 4-link serially connected fixed-base robot from $\text{LINK} = \{FB, L, L, L\}$ and $\text{JOINT} = \{R, R, R\}$ with a given kinematic graph G_3 shown in Fig.4.11, where FB stands for a fixed base. We assume that there is only one way to attach a revolute joint to the base. Furthermore, we restrict each joint to be located at each end of the prism. Without this constraint, there will be 12 distinct patterns as shown in Fig. 4.2. But we see there are only 7 distinct assembly patterns satisfying this condition. Hence, the number of possible patterns on link 2 and 3 are 7, while there are 2 patterns on link 4. Totally, there are at most $N_{G_3} = 1 \times 7 \times 7 \times 2 = 98$ distinct configurations. They are shown in Fig. 4.12 to 4.13. Since G_3 has no symmetry, the automorphism group of G_3 contains only the identity element. The actual number of distinct configurations achieves the upper bound N_{G_3} . If we do not pay attention to equivalent assembly patterns caused by the link symmetry, there are 25 patterns on link 2 and 3 each. (Each of the two joints can be attached to either one of the 5 ports at the end of a prism, so there are 25 patterns for each link.) Totally, there will be $1 \times 25^2 \times 2 = 3125$ constructions. ■

In general, for an n -DOF fixed base robot using n R-joints and n prisms with a fixed base with above constraint, there are $N_G = 2 \times 7^{n-1}$ distinct assembly configurations. Table 4.3 shows the comparison between our algorithm and brute force enumeration of the total number of assembly configurations of this n -DOF fixed base serial robot.

| DOF | RobotEnumerate | Brute Force |
|-----|----------------|-------------|
| 2 | 14 | 125 |
| 3 | 98 | 3125 |
| 4 | 686 | 78125 |
| 5 | 4802 | 1953125 |

Table 4.3: Comparison between two algorithms

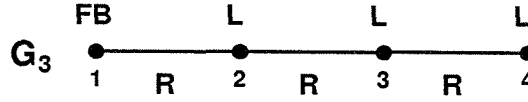


Figure 4.11: The graph of a 3-DOF serial modular robot

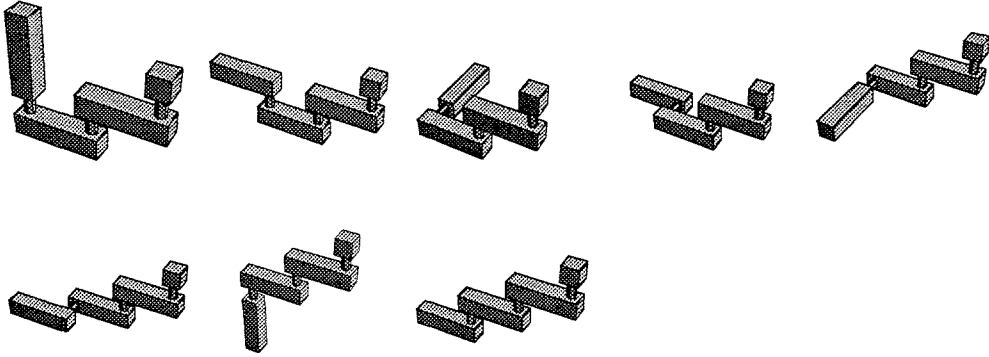


Figure 4.12: Distinct assembly configurations of the 3-DOF robot

4.3.3. Computational Complexity Issues

Let us briefly consider the computational complexity of this algorithm. The tree generation algorithm in step 1 is constant time for a given number of vertices. To find the automorphism group of a graph in step 2 requires an exhaustive search on isomorphisms of the graphs. Backtrack is basically an exponential time search algorithm [89]. In step 3, the time to compute distinct assignments on a graph G_i under $\mathcal{H}(G_i)$ is $O(K^2)$, where K is the number of distinct assignments, since we are using the OrbitEnumerate algorithm. In step 4, we perform at most $|\mathcal{H}(G)|$ checks to generate the automorphism group, $\mathcal{H}(\tilde{G})$, of the specialized graph \tilde{G} .

In step 5-(a), the time to generate distinct assembly pattern on every link v_i is $O(N_{v_i}^2)$, where N_{v_i} is the number of distinct assembly patterns. Since some links in a robot will be the same, these patterns can be calculated in advance and stored in a look-up table to save computation time. It is unnecessary to compute distinct assemblies for links having identical joint patterns repeatedly.

Step 5-(b) gives an upper bound, N_{G_i} , on the number of distinct configurations for a given graph G_i . N_{G_i} equals the product of the number of distinct joint patterns on

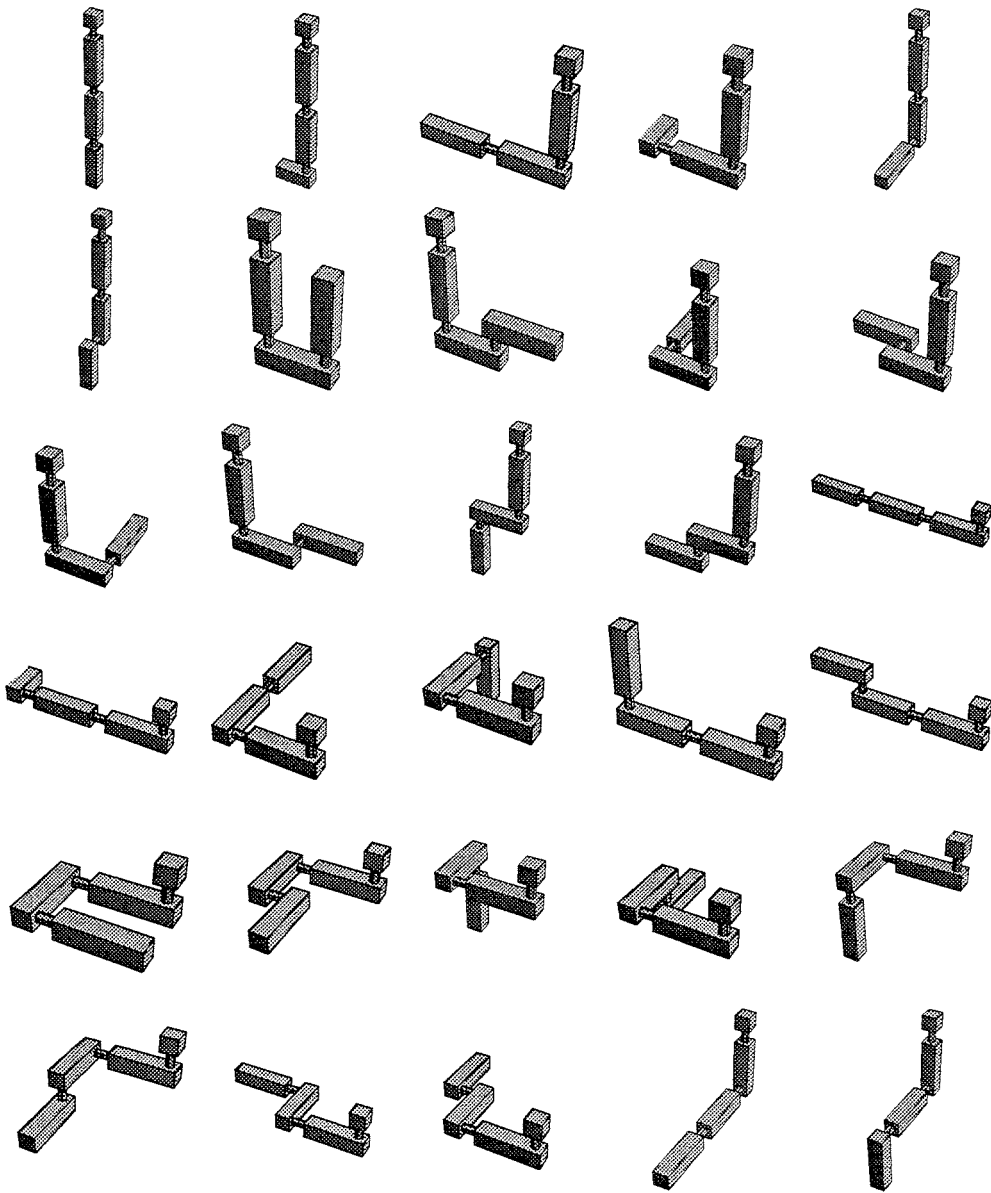


Figure 4.13: 3-DOF fixed base robot example (continued)

every link of G_i . Owing to graph symmetry, the actual distinct configurations is always less than or equal to N_{G_i} . The upper bound will be achieved only if the graph has no symmetry, i.e., the automorphism group defining the graph symmetry contains only the identity element. The sum of N_{G_i} 's for all non-isomorphic graphs gives the upper bound on the total number of distinct n -link tree modular robot assembly configurations.

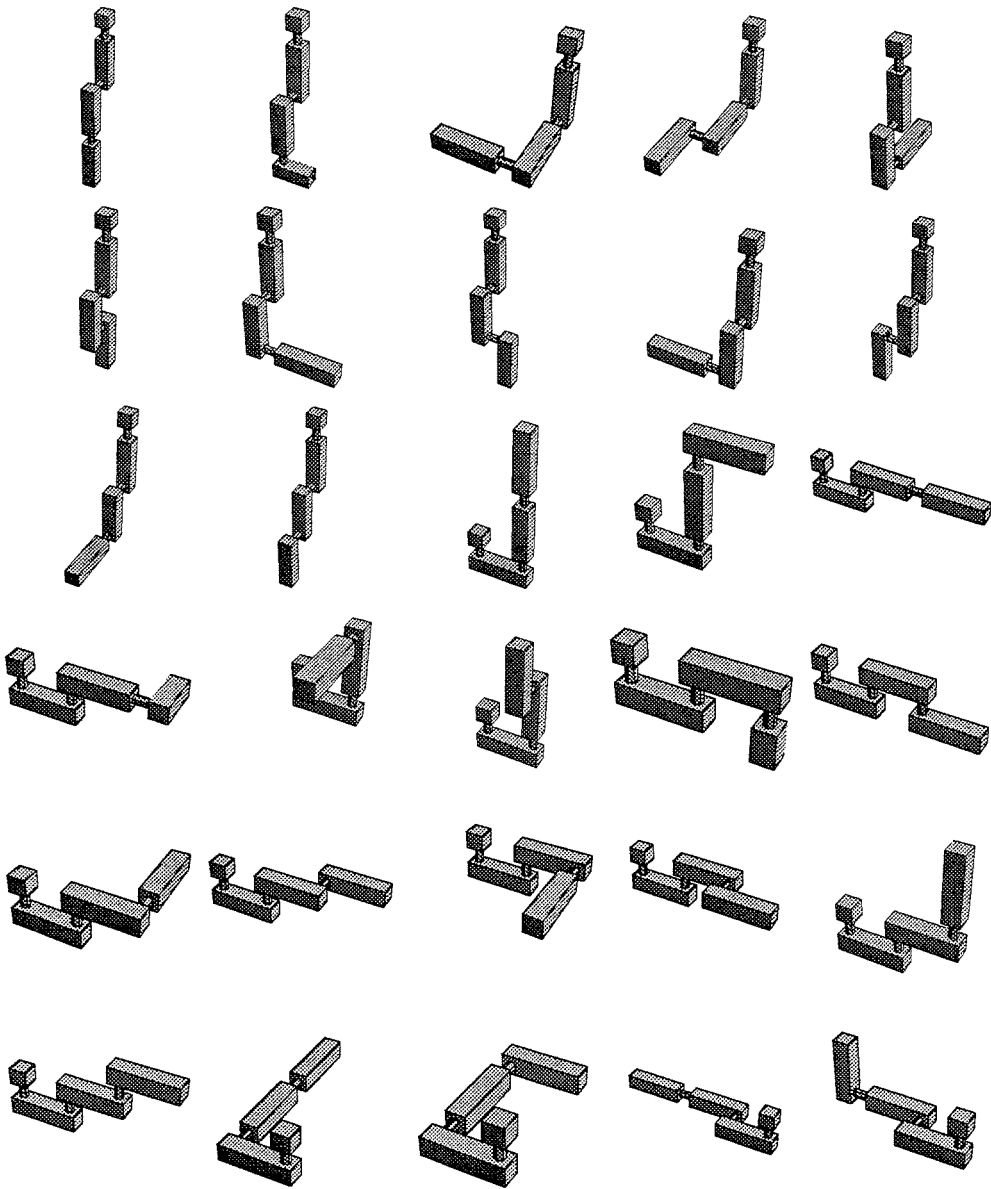


Figure 4.14: 3-DOF fixed base robot example (continued)

Step 5-(c) checks the automorphisms of robot assembly configurations generated by the previous step in a pairwise fashion if the graph exhibits symmetry. Since step 5-(b) generates N_{G_i} AIMs, we have to check $\binom{N_{G_i}}{2}$ pairs of AIMs. Generally speaking, $O(N_{G_i}^2)$ time is needed to perform checks on graph G_i .

Note that computationally costly graph isomorphism checks on labeled and specialized

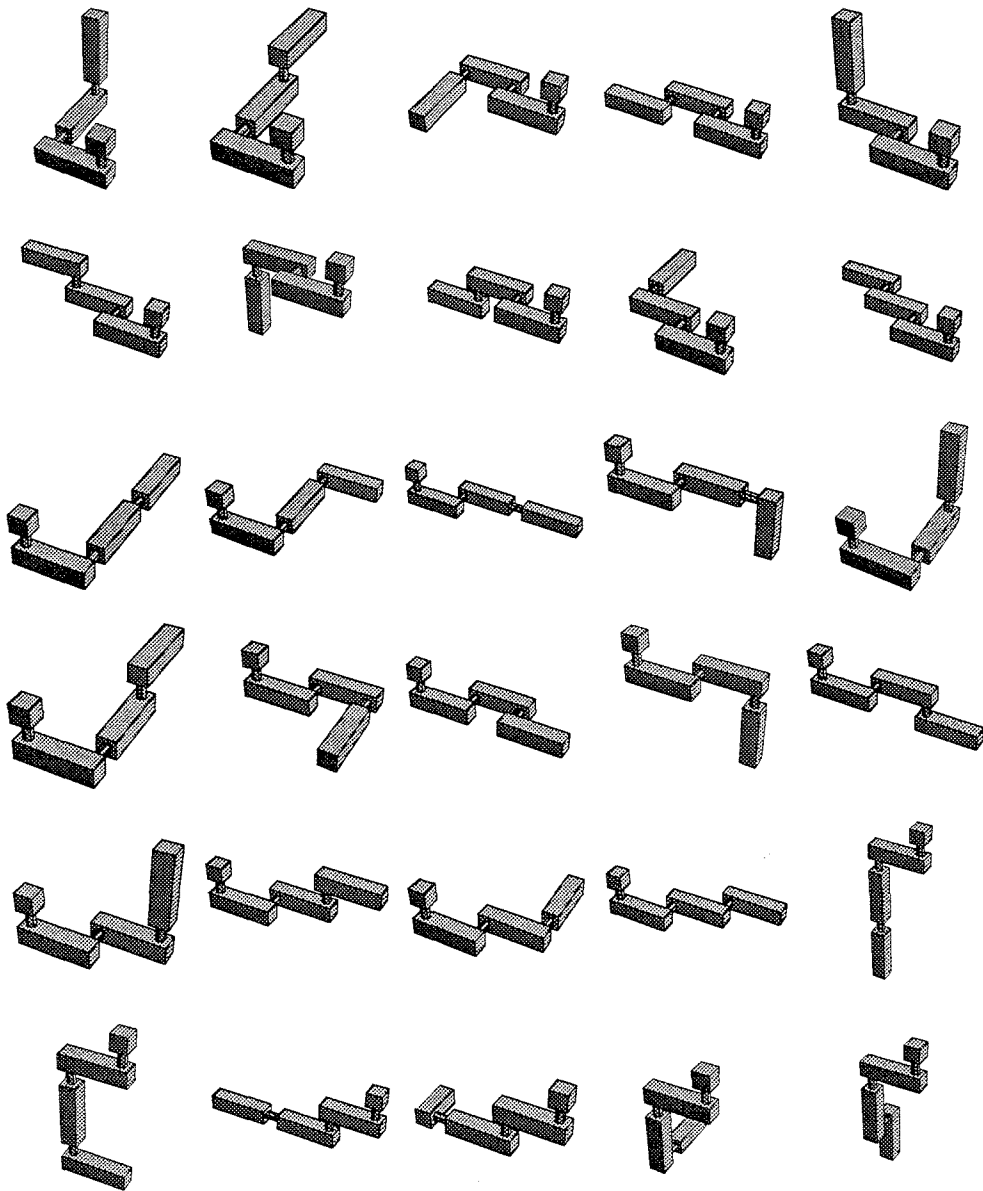


Figure 4.15: 3-DOF fixed base robot example (continued)

graphs are unnecessary in the enumeration process because we separate the generation of non-isomorphic graphs procedure at the beginning. There is no known polynomial time algorithm to check graph isomorphisms [89]. Only when two AIMs are given without any previous knowledge of the underlying kinematic graphs, an isomorphism test is needed.

This discussion points out several features of the algorithm. First, its computational complexity depends on the properties of the link symmetry groups and the class of tree-like structures one is considering. It is thus difficult to give a precise bound on the computational complexity of the entire algorithm. Second, the computations are structured so as to avoid computationally expensive steps, such as graph isomorphism checking. Third, the reasonable computational complexity of the algorithm (and the examples of Section 4.3.2) implies that for almost any conceivable application, it is much more efficient to enumerate the non-isomorphic geometries with this algorithm, rather than using a brute enumeration process.

4.4. Closed-Loop Construction Enumeration

From the analysis of linkage mechanism, we know the kinematic constraints and number constraints on closed-loop mechanisms. The number and type of close-loop construction depends on the actual module designs, and are not appropriate to be determined by any graph enumeration schemes. One feasible way to incorporate closed-loop module constructions in a modular robot system is to treat each one as an individual link. Because the module geometry is designed in advance, the number of non-isomorphic close-loop configurations are limited and can be found manually. The symmetric rotation group for each configuration can be defined. Each configuration is then expressed as a single vertex in the kinematic graph of the modular robot.

Example 4.19: Consider using four revolute joints and four square prisms to construct a closed-loop module robot subassembly. The eight non-isomorphic closed-loop constructions are shown in Fig. 4.16. Because of the additional constraint on closing the kinematic chain, these closed-loop assemblies are found manually. ■

4.5. Discussion

One of our basic assumptions on our module models is its symmetric shape. In an extreme case, the link module has no symmetry and its symmetric rotation group has

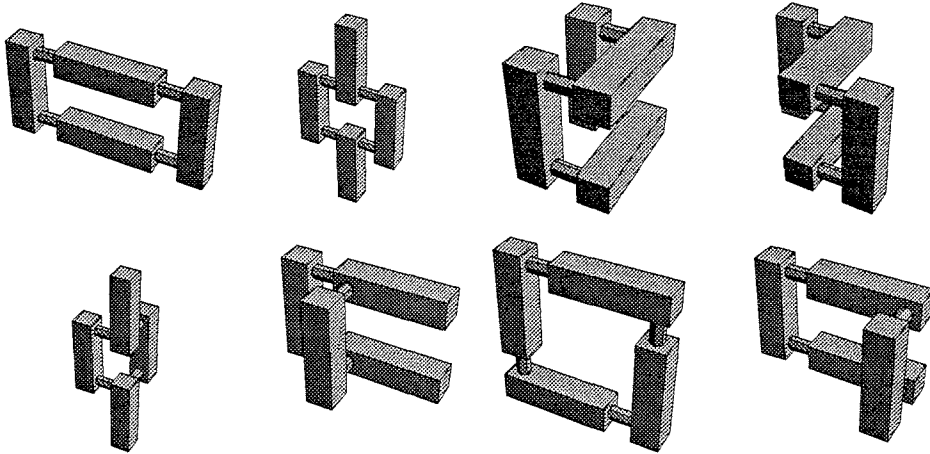


Figure 4.16: Eight non-isomorphic closed-loop configurations

only one element—the identity element. The enumeration problem still follows the same procedure outlined in Section 7.4.3.2. Without symmetry, the process of finding distinct joint assemblies becomes equivalent to permuting the joints on the link ports. In this case, there is no need to use the orbit enumeration algorithm. However, if a link module has no symmetry, the interchangeability of this module will be greatly reduced.

In summary, this chapter demonstrated a method to enumerate distinct assembly configurations of a tree-like modular robot from a set of specified modules. Distinct modular robots have distinct kinematic properties and, hence, different functionalities. An algorithm is proposed to enumerate joint assembly patterns on a link. Then a novel class of assembly incidence matrices (AIMs) is defined to represent robot assembly configurations. A two-stage equivalence relation based on graph isomorphism and link symmetry is introduced for AIMs. Inequivalent AIMs are distinct assembly configurations having different robot kinematics. These representations and equivalences form the basis for our algorithm. As demonstrated in the examples, this algorithm greatly improves the efficiency of the enumeration process as compared to brute force enumeration.

Chapter 5

Modular Robot Kinematics

This chapter studies three issues in modular robot kinematics: forward kinematics, inverse kinematics, and the kinematic equivalence of distinct (or non-isomorphic) robot assembly configurations. The terms forward and inverse kinematics assume their usual meaning. In the context of modular robots, the notion of kinematic equivalence regards modular robots having distinct AIMS but possessing identical kinematic properties, such as workspace shapes and volumes, and joint singularities.

Several kinematic modeling techniques for traditional fixed configuration robot manipulators have been developed: the Denavit-Hartenburg (D-H) parameterization method [21]; the Product-of-Exponential Formulas (POE) [9,75,78,79,80], and the Sheth-Uicker notations [88]. However, for modular robots, these kinematic modeling methods are not so useful. Furthermore, the generic topology of a modular robot has a tree-structure instead of a serial chain, which is implied in the above modeling methods.

Researchers in CMU [46] and University of Toronto [5] have proposed methods, based on the D-H kinematic modeling scheme, to automatically generate modular robot forward kinematics. The method of [46] used two sets of coordinate frames to describe a modular robot: *modular frames* for individual modules showing kinematic parameters of links and joints of RMMS, and *D-H frames* defined with D-H notations. With a given sequence of modules, a conversion algorithm from the modular frames to D-H

frames was employed to automatically obtain the D-H parameters of an entire robot. However, only revolute joints were considered in the kinematic conversion scheme. The work of [5] generalized this modeling technique to include prismatic joints and considered the possibility of multiple attachment of joints on one link module by defining *input/output frames* on every link.

In this chapter, we propose a scheme, based on the product-of-exponentials, to generate tree-structure robot forward kinematics from a given AIM automatically. In this scheme, a robot assembly is represented by its intrinsic properties, i.e., the location and the type of the joint axes, which are conveniently derived from an AIM by a set of *port conversion tables* and *initial position tables*. Furthermore, these intrinsic properties facilitate the analysis on the kinematic performance of a robot. Most important of all, this scheme is completely general for tree-structured robot topologies that are not considered in previous work.

Because a modular robot has no fixed assembly configuration and the number of DOF in the robot varies, it is very difficult to find closed form inverse kinematic solutions. Therefore, we adopt a numerical inverse kinematics technique proposed by Khosla, Neuman, and Prinz [47] for the reconfigurable robotic system. This numerical method is based on Newton-Ralphson iteration.

From task point of view, the robot's kinematic performance is a major concern. In practice, there is a small subset of distinct AIMs that possess identical kinematic properties and are therefore functionally equivalent. A kinematic equivalence relation based on the twist of the joints will be defined on AIMs for classification.

This chapter is organized as follows. Section 5.1 studies the forward kinematics of modular robots. An algorithm to automatically generate tree-structured robot forward kinematics from an AIM is proposed. Section 5.2 defines kinematic equivalence on AIMs. Section 5.3 discusses the numerical inverse kinematic scheme used by modular robots.

5.1. Forward Kinematics

Traditionally, the displacement of a rigid body can be expressed as the position and orientation of a body fixed coordinate frame relative to a world fixed reference frame. In order to distinguish this displacement, which is often termed a “configuration,” g , from the assembly configuration of a modular robot, we call g a “position/orientation” instead. For spatial motion, g can be written as a pair $(R, \mathbf{p}) \in SO(3) \times \mathbb{R}^3$, where R is the orientation of the body frame and \mathbf{p} is the displacement from the origin of the world frame to the origin of the body frame. We also use the notation $SE(3) \equiv SO(3) \times \mathbb{R}^3$. g can also be written as a 4×4 homogeneous matrix:

$$g = \begin{pmatrix} R & \mathbf{p} \\ 0 & 1 \end{pmatrix} \quad (5.1)$$

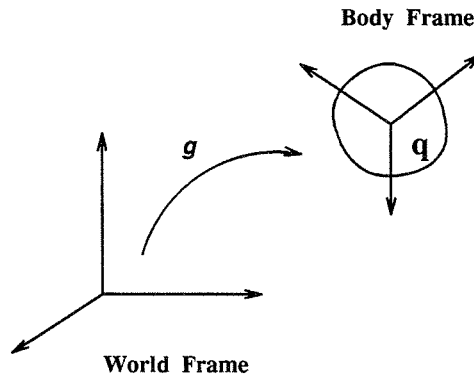


Figure 5.1: A rigid body displacement g

Suppose $\mathbf{q} \in \mathbb{R}^3$ is a point on the body relative to the world frame as shown in Fig. 5.1. The action of g on the point \mathbf{q} in the body defines the new location of \mathbf{q} as

$$g(\mathbf{q}) = R\mathbf{q} + \mathbf{p}. \quad (5.2)$$

This action can also be written in the 4×4 matrix form by representing the point \mathbf{q} as a 4-vector, $\tilde{\mathbf{q}} = (\mathbf{q}, 1)$:

$$g(\tilde{\mathbf{q}}) = \begin{pmatrix} g(\mathbf{q}) \\ 1 \end{pmatrix} = \begin{pmatrix} R & \mathbf{p} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{q} \\ 1 \end{pmatrix} = \begin{pmatrix} R\mathbf{q} + \mathbf{p} \\ 1 \end{pmatrix}. \quad (5.3)$$

The motion of the body is a time parameterized curve, $g(t)$. The generalized velocity of the rigid body motion written in the world reference frame can be expressed as

$$\hat{\xi} = \dot{g}g^{-1} = \begin{pmatrix} \dot{R}R^T & \dot{\mathbf{p}} - \dot{R}R^T\mathbf{p} \\ 0 & 0 \end{pmatrix}. \quad (5.4)$$

This generalized velocity, $\hat{\xi}$, is called a twist [69]. It can also be written as

$$\hat{\xi} = \begin{pmatrix} S(\mathbf{w}) & \mathbf{v} \\ 0 & 0 \end{pmatrix}, \quad (5.5)$$

where $\mathbf{w} = (w_x, w_y, w_z) \in \mathbb{R}^3$ and $\mathbf{v} \in \mathbb{R}^3$. S is the skew symmetric matrix associated with \mathbf{w} , and

$$S(\mathbf{w}) = \begin{pmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{pmatrix}. \quad (5.6)$$

The vector

$$\xi = \begin{pmatrix} \mathbf{v} \\ \mathbf{w} \end{pmatrix} \quad (5.7)$$

is called the twist coordinate of $\hat{\xi}$ and represents the rotational and translational velocity of an object viewed in the world frame. The forward kinematics of an n -DOF robot written in POE can be expressed as follows [9].

$$f(\theta_1, \dots, \theta_n) = e^{\hat{\xi}_1\theta_1} e^{\hat{\xi}_2\theta_2} \dots e^{\hat{\xi}_n\theta_n} g(0) \quad (5.8)$$

where θ_i is the i^{th} joint displacement, $\hat{\xi}_i$ is the twist of the i^{th} joint axis, $g(0)$ is the initial position/orientation of the end-effector frame, and $f(\theta_1, \dots, \theta_n) \in SE(3)$. The map, f , is called a *forward kinematic map*. It defines the position/orientation of the end-effector frame with respect to the world reference frame.

The twist coordinate ξ of a joint axis is expressed relative to the world reference frame. For an R-joint, $\xi_i = (\mathbf{v}, \mathbf{w})$, where \mathbf{w} is a unit vector in the direction of the joint axis and $\mathbf{v} = -\mathbf{w} \times \mathbf{q}$. The point \mathbf{q} is located on the joint axis. For a P-joint, $\xi = (\mathbf{v}, 0)$, where \mathbf{v} is a unit vector pointing in the direction of translation.

Note that in this formulation, the initial position/orientation of the end-effector frame, $g(0)$, can be chosen arbitrarily. If the world reference frame is chosen to be coincident

with the initial end-effector frame, then $g(0) = I$. The location of the joint axes, i.e., the twist coordinates, are all relative to the world frame. The order of the joints are expressed in the order of the matrix operation.

To overcome the problem of no previous knowledge of the module arrangement and robot topology in assembling a modular robot, the forward kinematics of a modular robot is defined in a modular fashion. First, a set of table, termed the *port conversion tables*, relating the number on the connecting ports and their actual locations on the link is introduced. Using this table, a joint twist can be found according to the joint assembly pattern or a row vector of the AIM (or eAIM). Secondly, from another set of tables, termed the *initial position tables*, relating elements in the AIM and the initial position of two connected links (or a dyad), the forward transformation between the two links can be derived. Finally, applying graph search techniques along with this forward transformation on a dyad, the forward kinematics of a general tree-structured robot can be obtained.

5.1.1. Single Link Kinematics

The twist coordinate of the joint axes relative to the link frame can be obtained from the joint assembly patterns by a mapping between the actual location and the number on the connecting ports. Suppose that the dimensions of all connecting interfaces are the same. The actual location of a connecting port can be associated with an imaginary line, termed a *connecting line*. Any joint attached to or detached from the port will follow along this imaginary line. This line becomes the axis of rotation when a R, C, or H-joint is attached; it describes the direction of translation as a P-joint is attached to the link.

A line in \mathbb{R}^3 has a 6-vector representation called *Plücker* coordinates, $L = (\mathbf{w}, \mathbf{q} \times \mathbf{w})$, where \mathbf{w} is a unit 3-vector in the direction of the line and $\mathbf{q} \in \mathbb{R}^3$ is any point on the line. Let an R-joint be connected to Port i , where $i \in \text{PORT}$. The connecting line is defined as follows.

Definition 5.1: A connecting line associated with Port i on a link is defined to be a line coincident with the joint axis of the R-joint when it is attached to Port i as shown in Fig. 5.2. The Plücker coordinate of the connecting line is denoted by $L_i = (\mathbf{w}_i, \mathbf{q}_i \times \mathbf{w}_i)$. The direction of L_i is pointing away from the link module.

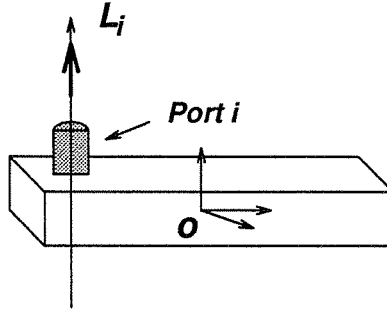


Figure 5.2: The connecting line associated with port i

Note that \mathbf{w} , the direction of L , defines the normal direction of the connecting port; $\mathbf{q} \times \mathbf{w}$, the location of L , defines the location of the port relative to the module frame. Although ports on the opposite faces of the link share the same line, their connecting lines are written in the two different representations of the same line, i.e., $(\pm\mathbf{w}, \pm\mathbf{q} \times \mathbf{w})$.

The index i associated with the connecting line L_i actually defines a mapping from the numbers on the ports to their associated connecting lines.

Definition 5.2: Let PORT be a port index set on a link. A *port conversion table* (PCT), \mathcal{T}_{pc} , is a 1-to-1 mapping such that it assigns a connecting line to a port, i.e.,

$$\mathcal{T}_{pc} : \text{PORT} \rightarrow \mathbb{R}^6 : i \mapsto L_i. \quad (5.9)$$

Different types of links have different port locations and, hence, different PCTs. Because the number on the ports can be arbitrarily defined, a PCT is not unique even on the same type of links.

For prism link modules, the link dimension related to the PCT is the distance between connecting lines of the two side ports, as shown in Fig. 2.6. Let $l = \text{PortSeparation}$, which is defined in Section 2.3. The connecting lines associated with the ports of a

cube module intersect at the center of symmetry (or the origin of the module frame) so the normal directions of the port become the major concern. Tables 5.1 and 5.2 demonstrate a set of port conversion tables for the prism link and the cube modules.

| Ports | Line Coordinates |
|-------|--------------------------|
| 1 | $(1, 0, 0, 0, l/2, 0)$ |
| 2 | $(1, 0, 0, 0, -l/2, 0)$ |
| 3 | $(0, 1, 0, -l/2, 0, 0)$ |
| 4 | $(0, 1, 0, l/2, 0, 0)$ |
| 5 | $(-1, 0, 0, 0, -l/2, 0)$ |
| 6 | $(-1, 0, 0, 0, l/2, 0)$ |
| 7 | $(0, -1, 0, l/2, 0, 0)$ |
| 8 | $(0, -1, 0, -l/2, 0, 0)$ |
| 9 | $(0, 0, 1, 0, 0, 0)$ |
| 10 | $(0, 0, 1, 0, 0, 0)$ |

Table 5.1: PCT for prism module (L)

| Ports | Line Coordinates |
|-------|-----------------------|
| 1 | $(1, 0, 0, 0, 0, 0)$ |
| 2 | $(0, 1, 0, 0, 0, 0)$ |
| 3 | $(-1, 0, 0, 0, 0, 0)$ |
| 4 | $(0, -1, 0, 0, 0, 0)$ |
| 5 | $(0, 0, 1, 0, 0, 0)$ |
| 6 | $(0, 0, -1, 0, 0, 0)$ |

Table 5.2: PCT for cubic box module (B)

To find the local representation of a joint axis twist in a modular robot or in a joint assembly pattern of a single link module, we perform the following procedure:

1. Apply a PCT, \mathcal{T}_{pc} , to the port numbers described in a row vector of an AIM, or in an assembly pattern, f , and find the corresponding connecting line.
2. Convert the Plücker coordinates to twist according to the type of joints. Let $L_i = (\mathbf{w}_i, \mathbf{v}_i)$ be the line associated with Port i . For an R-joint, the twist, $\xi_i = (\mathbf{v}_i, \mathbf{w}_i)$. For a P-joint, $\xi_i = (\mathbf{w}_i, 0)$.

Example 5.3: Suppose an assembly pattern on a prism module is

$$f = \begin{pmatrix} 1 & 2 & 3 & \cdots & 10 \\ R & P & 0 & \cdots & 0 \end{pmatrix}.$$

From Table 5.2, we get

$$\begin{aligned} L_1 &= (1, 0, 0, 0, l/2, 0) & \rightarrow & \quad \xi_1 = (0, l/2, 0, 1, 0, 0) \\ L_2 &= (1, 0, 0, 0, -l/2, 0) & \rightarrow & \quad \xi_2 = (1, 0, 0, 0, 0, 0) \end{aligned} \quad (5.10)$$

Example 5.4: The fourth row of $A(\tilde{G})$ in (4.6) is $(0, 0, 2, B)$, which says that joint e_3 is connected to link v_4 via Port 2. From the last row of $A(\tilde{G})$: $(C, C, R, 0)$, one knows that joint e_3 is an R-joint. Using Table 5.1, we obtain

$$L_2 = (0, 1, 0, 0, 0, 0) \quad \xi_2 = (0, 0, 0, 0, 1, 0). \quad (5.11)$$

5.1.2. Dyad Kinematics

Suppose an AIM, $A(\tilde{G})$, of an n -link tree-structured robot is given. Also assume that link v_i and link v_j are connected by joint e_k as shown in Fig. 5.3 and Fig. 5.4. Denoting the module frame on v_i by frame i and the position/orientation of frame j relative to frame i by T_{ij} , then by (5.8), the position of frame j under a joint displacement, θ_{ij} , can be written as

$$T_{ij}(\theta_{ij}) = e^{\hat{\xi}_{ij}\theta_{ij}} T_{ij}(0), \quad (5.12)$$

where $\hat{\xi}_{ij}$ is the twist coordinate of joint e_k relative to frame i , and $T_{ij}(0) \in SE(3)$ is the zero (or initial) position of frame j relative to frame i .

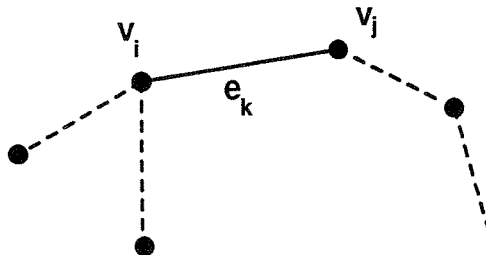


Figure 5.3: A schematic kinematic graph

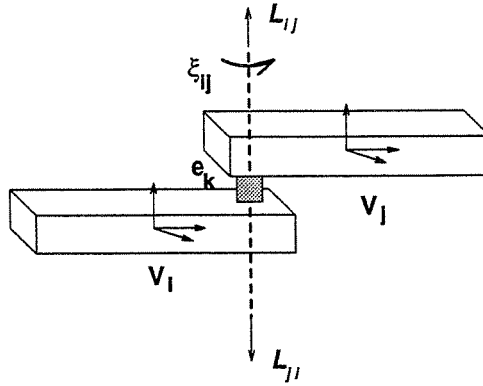


Figure 5.4: Part of the module assembly

Equation (5.12) defines the forward transformation of the dyad from link i to j . If the robot is serially connected and the links are marked from 1 to n consecutively, where the base link is denoted link 1 and the end link denoted link n , then the forward transformation from link 1 to n is

$$T_{1n} = T_{12} T_{23} \cdots T_{(n-1)n}. \quad (5.13)$$

If the robot has a tree structure, the forward transformation (5.12) can be applied recursively to find the position/orientation of all pendant links with respect to the base root link. Link v_i is the predecessor of link v_j because the direction of transformation is from link i to j . Section 5.1.3 will present an algorithm to compute the tree-structured robot kinematics.

There are five elements in the eAIM, $A(\tilde{G})$, related to the dyad of link i and j : $(a_{ik}, a_{in}, a_{jk}, a_{jn}, a_{(n+1)k})$, termed the *dyad vector*, where $a_{(n+1)k}$ represents the type of joint e_k ; a_{in} and a_{jn} represent the type of link v_i and v_j . Joint e_k is connected to Port a_{ik} and a_{jk} of link v_i and v_j respectively. Their locations in the eAIM is shown below:

$$A(\tilde{G}) = \begin{pmatrix} \vdots & \vdots \\ \cdots & a_{ik} & \cdots & a_{in} \\ \vdots & \vdots \\ \cdots & a_{jk} & \cdots & a_{jn} \\ \vdots & \vdots \\ \cdots & a_{(n+1)k} & \cdots & 0 \end{pmatrix} \quad (5.14)$$

If a hashed AIM, $A^*(\tilde{G})$, is employed instead of the eAIM, $A(\tilde{G})$, then two elements, (a_{ik}, a_{jk}) , suffice to express the dyad vector because the types of joints and links have been coded in these elements already.

In order to generate robot kinematics automatically, a relation between the invariant quantities, e.g., $\hat{\xi}_{ij}$ and $T_{ij}(0)$, of the forward transformation (5.12) and the dyad vector of link i and j must be established. θ_{ij} is a variable to be determined by the input.

Determination of ξ_{ij}

Just as we did in Example 5.4, applying the PCT of link i to the element a_{ik} of $A(\tilde{G})$, the connecting line L_{ij} can be determined. The twist ξ_{ij} of joint e_k can be obtained by the procedure mentioned in the last section.

Determination of $T_{ij}(0)$

Instead of setting up a table that assigns dyad vectors to $T_{ij}(0)$ directly, a table called the *initial position table* (IPT) which assigns dyad vectors to the initial z-axis of frame j relative to frame i is used. The derivation of $T_{ij}(0)$ from the IPT is discussed in the following.

Definition 5.5: The *initial position table* (IPT) for a dyad of link v_i and v_j of Type a_{in} and a_{jn} connected by a joint of Type $a_{(n+1)k}$, where $a_{in}, a_{jn} \in \tilde{V}$ and $a_{(n+1)k} \in \tilde{E}$, is a 1-to-1 mapping,

$$\mathcal{T}_{ip} : \text{PORT}(a_{in}) \times \text{PORT}(a_{jn}) \times \tilde{E} \rightarrow \mathbb{R}^6 : (a_{ik}, a_{jk}) \mapsto L_z, \quad (5.15)$$

where L_z is the Plücker coordinate of the z-axis of frame j relative to frame i .

As shown in Fig. 5.4, the dyad of link i and j share the same connecting line having different expressions in frame i and j respectively. The Plücker coordinate of the connecting line L_{ij} with respect to frame i is obtained by using a PCT as mentioned above. Similarly, applying the PCT of link j to a_{jk} , we get L_{ji} , the same connecting line, but written in frame j .

Fig. 5.4 also shows the spatial relation of module frames i and j and the joint axis of e_k . Let $T(0) \equiv T_{ij}(0)$, $L_{ij} = (\mathbf{w}_i, \mathbf{v}_i)$, and $L_{ji} = (\mathbf{w}_j, \mathbf{v}_j)$. The following condition must be satisfied:

$$-\begin{pmatrix} \mathbf{v}_i \\ \mathbf{w}_i \end{pmatrix} = \text{Ad}_{T(0)} \begin{pmatrix} \mathbf{v}_j \\ \mathbf{w}_j \end{pmatrix} \quad (5.16)$$

where $T(0)$ is a 4×4 homogeneous matrix:

$$T(0) = \begin{pmatrix} R_{ij} & \mathbf{d}_{ij} \\ \bar{0} & 1 \end{pmatrix} \quad (5.17)$$

$R_{ij} \in SO(3)$ and $\mathbf{d}_{ij} \in \mathbb{R}^3$. $\text{Ad}_{T(0)}$ is 6×6 adjoint transformation matrix for $T(0) \in SE(3)$ [69] of the form:

$$\text{Ad}_{T(0)} = \begin{pmatrix} R_{ij} & S(\mathbf{d}_{ij})R_{ij} \\ 0 & R_{ij} \end{pmatrix} \quad (5.18)$$

The minus sign is added on the left hand side of (5.16) because the port connecting lines are pointing outward of the link and L_{ij} and L_{ji} are in opposite directions.

In equation (5.16), L_{ij} and L_{ji} are known quantities; $T(0)$ or $\text{Ad}_{T(0)}$ remains to be determined. Equation (5.16) can be written as two sets of three equations:

$$-\mathbf{w}_i = R_{ij}\mathbf{w}_j \quad (5.19)$$

$$-\mathbf{v}_i = R_{ij}\mathbf{v}_j + S(\mathbf{d}_{ij})R_{ij}\mathbf{w}_j \quad (5.20)$$

Because \mathbf{w}_i and \mathbf{w}_j are unit vectors and $R_{ij} \in SO(3)$, (5.19) and (5.20) have an infinity solution. That is, the initial position $T(0)$ cannot be determined directly by (5.16). In order to find a unique solution to (5.16) for a given pair of L_{ij} and L_{ji} , an additional constraint equation is required.

With a predetermined initial $L_z = (\mathbf{W}_{ij}, \mathbf{v}_{ij})$ (the z-axis of frame j relative to frame i), we have the following:

$$\mathbf{w}_{ij} = R_{ij} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (5.21)$$

Using (5.19) and (5.21), one can solve for R_{ij} . Substituting R_{ij} into (5.20), \mathbf{d}_{ij} can be solved. Then the initial position $T(0)$ can be obtained.

Note that if the rotational displacement R_{ij} is in the direction of the joint axis, \mathbf{w}_i , R_{ij} cannot be solved from (5.19). In this case, we solve for R_{ij} by first solving for $T_{ji}(0)$, the initial position of link j relative to link i . Let $T(0) \equiv T_{ji}(0)$ and use equations (5.16) to (5.21) by switching the indices i and j , then $T_{ij}(0) = T_{ji}^{-1}(0)$. If $T_{ij}(0)$ cannot be found in this way, the joint axis must be coincident with the z-axis of both links as shown in Fig. 5.5. In this situation, R_{ij} is set to I and \mathbf{d}_{ij} is determined by the type of the two connected links.

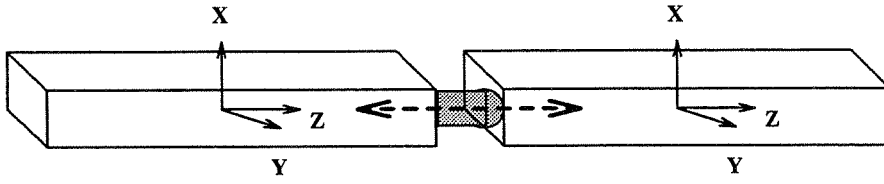


Figure 5.5: Situation for indeterminate R_{ij}

For an R-joint, the orientation of L_z of link j can be chosen arbitrarily. Usually the initial L_z is kept parallel or perpendicular to any one of the axes of Frame i . For a P-joint, L_z is determined by the extended feature ORIENT because the relative orientation of the two links is fixed.

The complete procedure to determine the forward transformation of the dyad of link i and j is as follows.

1. Locate elements in the AIM related to the dyad of link i and j , i.e., the dyad vector $(a_{ik}, a_{in}, a_{jk}, a_{jn}, a_{(n+1)k})$.
2. Find corresponding PCTs on link i and j and apply them to a_{ik} and a_{jk} to find the coordinate of the connecting lines, L_{ij} and L_{ji} respectively.
3. Apply IPT on the dyad vector, $(a_{ik}, a_{in}, a_{jk}, a_{jn}, a_{(n+1)k})$, to find the initial position, L_z , of the z-axis of frame j .
4. Use equations (5.19), (5.20), and (5.21) to solve for $T_{ij}(0)$.
5. If $T_{ij}(0)$ is indeterminate, switch the indices of i and j from (5.16) to (5.21), go back to Step 3, and solve for $T_{ji}(0)$. Then $T_{ij}(0) = T_{ji}^{-1}(0)$.

6. If $T_{ij}(0)$ cannot be found either way, then set its rotation matrix $R_{ij} = I$. \mathbf{d}_{ij} is the distance between the origin of frame i and j , which is determined by the types of the two connected links and the joint.

Since there are two different types of link modules in our module set, there are four possible combinations for their connection. For demonstration purpose, Tables 5.3 to 5.6 list all IPTs, where $j = \text{JointLength}$, $w = \text{LinkWidth}$ for prism modules, and $h = \text{LinkHeight}$ for cube modules.

Example 5.6: Consider the AIM of 3-DOF robot using three identical prism link modules and three R-joints:

$$A(\tilde{G}) = \begin{pmatrix} 1 & 0 & 0 & FB \\ 1 & 6 & 0 & L \\ 0 & 1 & 8 & L \\ 0 & 0 & 4 & L \\ R & R & R & 0 \end{pmatrix} \quad (5.22)$$

Now we use the above procedure to find the forward transformation from link 2 to link 3, i.e., T_{23} . The dyad vector for link 2 and 3 is: $(a_{22}, a_{24}, a_{32}, a_{34}, a_{52})$. From Table 5.1, the connecting line of port 6 on link 2 is $L_{23} = (-1, 0, 0, 0, l/2, 0)$. Similarly, $L_{32} = (1, 0, 0, 0, l/2, 0)$. Since the forward transform is from port 6 on link 2 to port 1 on link 3 and both links are prism modules, from Table 5.4, the z-axis of frame j is $L_z = (0, 0, 1, 0, j + w, 0)$. Substitute L_{23} , L_{32} , and L_z into (5.19), (5.20), and (5.21), and the initial position of frame j is

$$T_{23}(0) = \begin{pmatrix} 1 & 0 & 0 & -j - w \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (5.23)$$

The joint twist $\xi_{23} = (0, l/2, 0, -1, 0, 0)$. The forward transformation becomes

$$T_{23}(\theta_{23}) = e^{\theta_{23}\xi_{23}} T_{23}(0). \blacksquare$$

| Cube — Cube | | |
|-------------|---------|-----------------------------|
| Port 1 | Port 2 | P_z |
| 1 | 1,2,3,4 | $(0, 0, 1, 0, -(h + j), 0)$ |
| | 5 | $(-1, 0, 0, 0, 0, 0)$ |
| | 6 | $(1, 0, 0, 0, 0, 0)$ |
| 2 | 1,2,3,4 | $(0, 0, 1, 0, h + j, 0)$ |
| | 5 | $(0, -1, 0, 0, 0, 0)$ |
| | 6 | $(0, 1, 0, 0, 0, 0)$ |
| 3 | 1,2,3,4 | $(0, 0, 1, 0, h + j, 0)$ |
| | 5 | $(1, 0, 0, 0, 0, 0)$ |
| | 6 | $(-1, 0, 0, 0, 0, 0)$ |
| 4 | 1,2,3,4 | $(0, 0, 1, -(h + j), 0, 0)$ |
| | 5 | $(0, 1, 0, 0, 0, 0)$ |
| | 6 | $(0, -1, 0, 0, 0, 0)$ |
| 5 | 1,2,3,4 | $(1, 0, 0, 0, h + j, 0)$ |
| | 5 | $(0, 0, -1, 0, 0, 0)$ |
| | 6 | $(0, 0, 1, 0, 0, 0)$ |
| 6 | 1,2,3,4 | $(1, 0, 0, 0, -(h + j), 0)$ |
| | 5 | $(0, 0, 1, 0, 0, 0)$ |
| | 6 | $(0, 0, -1, 0, 0, 0)$ |

Table 5.3: IPT for Cube — Cube

5.1.3. Tree Robot Forward Kinematics Algorithm

Based on the dyad kinematics defined by (5.12), we propose an algorithm which can generate forward kinematics of a tree-structured robot from a given assembly configuration automatically. It is termed `TreeRobotKinematics`. The forward kinematics of a tree robot is defined to be the transformations from the base link to all pendant links. A serial robot is a tree structure without any branches. The links and joints are arranged in sequential order from the base to the end-effector or vice versa, so the transformation matrices of all intermediate links are obtained in order. However, the links and joints in a tree-structure robot do not have such ordering scheme; the order of the links is determined by tree-traversing algorithms. There are two frequently used traversing algorithms for graphs: Breath-First-Search (BFS) and Depth-First-Search (DFS) methods [89]. Either one can be employed for tree robot kinematics. Here we use the BFS approach because of its ease of implementation.

| Prism — Prism | | |
|---------------|-----------------|---|
| Port 1 | Port 2 | P_z |
| 1 | 1,3,5,7 | $(0, 0, -1, 0, j + w, 0)$ |
| | 2,4,6,8 | $(0, 0, 1, 0, -(j + w), 0)$ |
| | 9 | $(-1, 0, 0, 0, -l/2, 0)$ |
| | 10 | $(1, 0, 0, 0, l/2, 0)$ |
| 2 | 1,3,5,7 | $(0, 0, 1, 0, -(j + w), 0)$ |
| | 2,4,6,8 | $(0, 0, -1, 0, j + w, 0)$ |
| | 9 | $(-1, 0, 0, 0, l/2, 0)$ |
| | 10 | $(1, 0, 0, 0, -l/2, 0)$ |
| 3 | 1,3,5,7 | $(0, 0, -1, -(j + w), 0, 0)$ |
| | 2,4,6,8 | $(0, 0, 1, j + w, 0, 0)$ |
| | 9 | $(0, -1, 0, l/2, 0, 0)$ |
| | 10 | $(0, 1, 0, -l/2, 0, 0)$ |
| 4 | 1,3,5,7 | $(0, 0, 1, j + w, 0, 0)$ |
| | 2,4,6,8 | $(0, 0, -1, -(j + w), 0, 0)$ |
| | 9 | $(0, -1, 0, -l/2, 0, 0)$ |
| | 10 | $(0, 1, 0, l/2, 0, 0)$ |
| 5 | 1,3,5,7 | $(0, 0, -1, 0, -(j + w), 0)$ |
| | 2,4,6,8 | $(0, 0, 1, 0, j + w, 0)$ |
| | 9 | $(1, 0, 0, 0, l/2, 0)$ |
| | 10 | $(-1, 0, 0, 0, -l/2, 0)$ |
| 6 | 1,3,5,7 | $(0, 0, 1, 0, j + w, 0)$ |
| | 2,4,6,8 | $(0, 0, -1, 0, -(j + w), 0)$ |
| | 9 | $(1, 0, 0, 0, -l/2, 0)$ |
| | 10 | $(-1, 0, 0, 0, l/2, 0)$ |
| 7 | 1,3,5,7 | $(0, 0, -1, j + w, 0, 0)$ |
| | 2,4,6,8 | $(0, 0, 1, -(j + w), 0, 0)$ |
| | 9 | $(0, 1, 0, -l/2, 0, 0)$ |
| | 10 | $(0, -1, 0, l/2, 0, 0)$ |
| 8 | 1,3,5,7 | $(0, 0, 1, -(j + w), 0, 0)$ |
| | 2,4,6,8 | $(0, 0, -1, j + w, 0, 0)$ |
| | 9 | $(0, 1, 0, l/2, 0, 0)$ |
| | 10 | $(0, -1, 0, -l/2, 0, 0)$ |
| 9 | 1,2,3,4,5,6,7,8 | $(1, 0, 0, 0, l/2 + j + (w + h)/2, 0)$ |
| | 9 | $(0, 0, -1, 0, 0, 0)$ |
| | 10 | $(0, 0, 1, 0, 0, 0)$ |
| 10 | 1,2,3,4,5,6,7,8 | $(-1, 0, 0, 0, l/2 + j + (w + h)/2, 0)$ |
| | 9 | $(0, 0, 1, 0, 0, 0)$ |
| | 10 | $(0, 0, -1, 0, 0, 0)$ |

Table 5.4: IPT for Prism — Prism

| Cube — Prism | | |
|--------------|-----------------|--------------------------------------|
| Port 1 | Port 2 | P_z |
| 1 | 1,3,5,7 | $(0, 0, -1, 0, j + (w + h)/2, 0)$ |
| | 2,4,6,8 | $(0, 0, 1, 0, -(j + (w + h)/2), 0)$ |
| | 9 | $(-1, 0, 0, 0, 0, 0)$ |
| | 10 | $(1, 0, 0, 0, 0, 0)$ |
| 2 | 1,3,5,7 | $(0, 0, -1, -(j + (w + h)/2), 0, 0)$ |
| | 2,4,6,8 | $(0, 0, 1, j + (w + h)/2, 0, 0)$ |
| | 9 | $(0, -1, 0, 0, 0, 0)$ |
| | 10 | $(0, 1, 0, 0, 0, 0)$ |
| 3 | 1,3,5,7 | $(0, 0, -1, 0, -(j + (w + h)/2), 0)$ |
| | 2,4,6,8 | $(0, 0, 1, 0, j + (w + h)/2, 0)$ |
| | 9 | $(1, 0, 0, 0, 0, 0)$ |
| | 10 | $(-1, 0, 0, 0, 0, 0)$ |
| 4 | 1,3,5,7 | $(0, 0, -1, j + (w + h)/2, 0, 0)$ |
| | 2,4,6,8 | $(0, 0, 1, -(j + (w + h)/2), 0, 0)$ |
| | 9 | $(0, 1, 0, 0, 0, 0)$ |
| | 10 | $(0, -1, 0, 0, 0, 0)$ |
| 5 | 1,2,3,4,5,6,7,8 | $(1, 0, 0, 0, j + (w + h)/2, 0)$ |
| | 9 | $(0, 0, -1, 0, 0, 0)$ |
| | 10 | $(0, 0, 1, 0, 0, 0)$ |
| 6 | 1,2,3,4,5,6,7,8 | $(-1, 0, 0, 0, j + (w + h)/2, 0)$ |
| | 9 | $(0, 0, 1, 0, 0, 0)$ |
| | 10 | $(0, 0, -1, 0, 0, 0)$ |

Table 5.5: IPT for Cube — Prism

This algorithm takes three inputs: a hashed AIM of a hybrid robot $A^*(\tilde{G})$, the base link (the root vertex) v_b , and a set of joint angles $\{\theta\}$. The output will be the forward transformation of all links relative to the base link frame under the joint displacements. Note that in ordinary BFS or DFS search algorithm, a graph is represented by an *adjacency-list* [18]. An adjacency-list of a graph G consists of an array of lists. The number of lists is equal to the number of vertices in G . The i^{th} list contains all the vertices adjacent to vertex i in G . In a modular robot application, the adjacency informations of the edges and vertices in the kinematic graph are required. Therefore, a hashed AIM is chosen as the input assembly configuration because of its compactness. A modified adjacency-list called an *edge-vertex adjacency-list* (EV-list) is introduced

| Prism — Cube | | |
|--------------|---------|---|
| Port 1 | Port 2 | P_z |
| 1 | 1,2,3,4 | $(0, 0, -1, 0, j + (w + h)/2, 0)$ |
| | 5 | $(-1, 0, 0, 0, -l/2, 0)$ |
| | 6 | $(1, 0, 0, 0, l/2, 0)$ |
| 2 | 1,2,3,4 | $(0, 0, 1, 0, -(j + (w + h)/2), 0)$ |
| | 5 | $(-1, 0, 0, 0, l/2, 0)$ |
| | 6 | $(1, 0, 0, 0, -l/2, 0)$ |
| 3 | 1,2,3,4 | $(0, 0, -1, -(j + (w + h)/2), 0, 0)$ |
| | 5 | $(0, -1, 0, l/2, 0, 0)$ |
| | 6 | $(0, 1, 0, -l/2, 0, 0)$ |
| 4 | 1,2,3,4 | $(0, 0, 1, j + (w + h)/2, 0, 0)$ |
| | 5 | $(0, -1, 0, -l/2, 0, 0)$ |
| | 6 | $(0, 1, 0, l/2, 0, 0)$ |
| 5 | 1,2,3,4 | $(0, 0, -1, 0, -(j + (w + h)/2), 0)$ |
| | 5 | $(1, 0, 0, 0, l/2, 0)$ |
| | 6 | $(-1, 0, 0, 0, -l/2, 0)$ |
| 6 | 1,2,3,4 | $(0, 0, 1, 0, j + (w + h)/2, 0)$ |
| | 5 | $(1, 0, 0, 0, -l/2, 0)$ |
| | 6 | $(-1, 0, 0, 0, l/2, 0)$ |
| 7 | 1,2,3,4 | $(0, 0, -1, j + (w + h)/2, 0, 0)$ |
| | 5 | $(0, 1, 0, -l/2, 0, 0)$ |
| | 6 | $(0, -1, 0, l/2, 0, 0)$ |
| 8 | 1,2,3,4 | $(0, 0, 1, -(j + (w + h)/2), 0, 0)$ |
| | 5 | $(0, 1, 0, l/2, 0, 0)$ |
| | 6 | $(0, -1, 0, -l/2, 0, 0)$ |
| 9 | 1,2,3,4 | $(1, 0, 0, 0, l/2 + j + (w + h)/2, 0)$ |
| | 5 | $(0, 0, -1, 0, 0, 0)$ |
| | 6 | $(0, 0, 1, 0, 0, 0)$ |
| 9 | 1,2,3,4 | $(-1, 0, 0, 0, l/2 + j + (w + h)/2, 0)$ |
| | 5 | $(0, 0, 1, 0, 0, 0)$ |
| | 6 | $(0, 0, -1, 0, 0, 0)$ |

Table 5.6: IPT for Prism — Cube

as an internal representation of the hashed AIM.

Definition 5.7: An edge-vertex adjacency-list of a graph $G = (V, E)$ is an array of $|V|$ lists. Entries in the i^{th} list are ordered pairs of the form: (i, j) , which are the indices of non-zero entries in the i^{th} row of the incidence matrix of the graph, $M(G)$, the AIM,

$A(G)$, or the hashed AIM, $A^*(\tilde{G})$. The ordered pair (i, j) is called an *adjacency-index*.

Example 5.8: The EV list of the hashed AIM $A^*(\tilde{G})$ in (4.14) is

$$\begin{aligned} \text{AdjList} = \{ \{(1, 1), (1, 2), (1, 3)\} \\ \{(2, 2)\}, \{(3, 1)\}, \{(4, 3)\} \}. \blacksquare \end{aligned} \quad (5.24)$$

Instead of manipulating the vertices, the BFS algorithm used by `TreeRobotKinematics` is modified to manipulate the adjacency indices, i.e., to traverse the non-zero entries in the hashed AIM. The algorithm, written in pseudo code, works as follows.

```

0  Procedure TreeRobotKinematics( $A^*(\tilde{G}), b, \{\theta\}$ )
1  Queue =  $\{b\}$ ;
2  None = NonzeroEntry( $A^*(\tilde{G})$ );
3  AdjList = AIMtoEVList( $A^*(\tilde{G})$ );
4  Unvisit = None;
5   $T_b = Id$ ;
6  TransList =  $\{T_b\}$ ;
7  While Queue  $\neq \emptyset$  do
8      {
9       $v = \text{First}(\text{Queue})$ ;
10     Queue = Rest(Queue);
11     For all  $x \in \text{AdjList}[v]$  do
12         {
13         If  $x \in \text{Unvisit}$  then  $child = \text{MatchEdge}(\text{None}, x)$ ;
14          $l = \text{First}(child)$ ;
15          $m = \text{Last}(child)$ ;
16          $T_l = \text{ForwardTransform}(a_x, a_{child}, T_v, \theta_m)$ ;
17         TransList = Append( $T_l$ , TransList);
18         Append( $l$ , Queue);
19         Unvisit = Delete( $x$ , Unvisit);
20         Unvisit = Delete( $child$ , Unvisit);
21         };
22     };
23  Return(TransList);

```

The base link (the root of the tree) is chosen arbitrarily. The integer b represents the label on the base link. Queue is the set of visited links (vertices). None stores the adjacency indices of $A^*(\tilde{G})$. AdjList contains the EV-list of $A^*(\tilde{G})$. Unvisit keeps the unvisited adjacency indices. AIMtoEVList is a function that transforms IM-like matrices into an EV-list representation. The reference frame is chosen at the

base frame, so that the 4×4 matrix T_b is the identity matrix. Line 9 takes out the first element, v , from Queue. For all adjacency indices x in list v of EVList, we do the following: If x is not visited, Line 13 finds its descendent adjacency index called *child*. The Match-Edge function performs the descendent finding routine. The first element of *child* contains the label of the descendent link. Line 16 determines the position/orientation of the descendent link frame relative to the base reference frame using Forward-Transform. The descendent link is then stored in Queue in Line 18. Lines 19 and 20 delete the visited adjacency indices x and *child* from Unvisit. Line 23 returns a list, TransList, of the position/orientations of all link modules relative to the world frame.

The function ForwardTransform calculates the kinematics for adjacent links. It takes the dyad vector, (a_{ik}, a_{jk}) , the current location of frame i relative to the world frame, T_{bi} , and a joint displacement, θ_{ij} . The output is the current location of frame j relative to the world frame, T_{bj} . This procedure works as follows.

```

0  Procedure ForwardTransform( $a_{ik}, a_{jk}, T_{bi}, \theta_{ij}$ )
1   $T_{ij}(0) = \mathcal{T}_{ip}(a_{ik}, a_{jk});$ 
2   $\xi_{ij} = \mathcal{T}_{pc}(a_{ik});$ 
3   $T_{ij}(\theta_{ij}) = e^{\xi_{ij}\theta_{ij}} T_{ij}(0);$ 
4   $T_{bj} = T_{bi} T_{ij}(\theta_{ij});$ 
5  Return( $T_{bj}$ );

```

Line 1 determines the initial position $T_{ij}(0)$ by applying IPT, \mathcal{T}_{ip} , to the dyad vector (a_{ik}, a_{jk}) . Note that a_{ik} and a_{jk} are coded integers containing the port and types of the links and the joint in the dyad. Line 2 converts a_{ik} to ξ_{ik} by using a PCT, \mathcal{T}_{pc} . Line 3 calculates the forward kinematics from frame i to j , using (5.12). Line 4 transforms the local representation of frame j back to the world reference frame. Line 5 returns T_{bj} .

Since TreeRobotKinematics provides locations of all links relative to the world frame, T_{bi} , the twist of joint e_k on any one of the robot links with respect to the world frame,

ξ_b^k , can be written as

$$\xi_b^k = Ad_{T_{b_i}} \xi_{ij}. \quad (5.25)$$

The spatial Jacobian, J^s , of a serial manipulator is a matrix whose columns are the twist coordinates of the joint axes relative to the world reference frame [69]. Therefore,

$$J^s = (\xi_b^1 \quad \xi_b^2 \quad \cdots \quad \xi_b^n). \quad (5.26)$$

This Jacobian relates the joint rate $\dot{\theta}$ with the generalized velocity (or the twist) of the end-effector, ξ_{ee} , by

$$\xi_{ee} = J^s \dot{\theta}, \quad (5.27)$$

where $\xi_{ee} = (\mathbf{v}, \mathbf{w})$. \mathbf{v} is the velocity of a point in the end-effector frame passing through the origin of the world frame as viewed in the world frame, and \mathbf{w} is the angular velocity of the end-effector viewed in the world frame. With a given AIM and a set of joint angles, the Jacobian of a serial-connected modular robot can be obtained automatically by using (5.25) and (5.27).

Other functional procedures that must traverse the tree structure of a modular robot will also possess the basic feature of `TreeRobotKinematics`. Additional functions, such as the module drawing routine used for simulation purposes, can be added right after Line 16 of `TreeRobotKinematics`, where the adjacent link kinematics is calculated.

5.1.4. Forward Kinematics Examples

Example 5.9: Suppose five prisms and four revolute joints are provided to build a quadruped like robot. Its kinematic graph \tilde{G} is shown in fig. 5.6. The hashed AIM $A^*(\tilde{G})$ is

$$A^*(\tilde{G}) = \begin{pmatrix} 8 & 0 & 0 & 0 \\ 20 & 8 & 4 & 24 \\ 0 & 20 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix} \quad (5.28)$$

The base module is Link v_2 . The joint angles relative to the base are $\{\theta_1, \theta_2, \theta_3, \theta_4\} = \{-\pi/4, -\pi/4, \pi/4, \pi/4\}$. The quadruped is shown in Fig. 5.7(a).

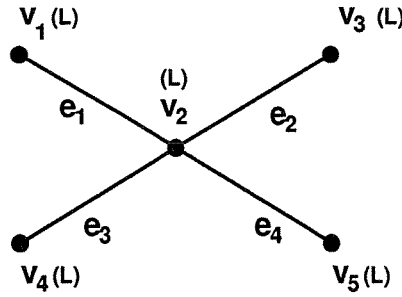


Figure 5.6: Kinematic graph of a quadruped \tilde{G}

Example 5.10: Three prisms and three revolute joint modules are given to build a fixed base 3-DOF serial manipulator arm. The eAIM is shown in (5.22). The hashed AIM is

$$A^*(\tilde{G}) = \begin{pmatrix} 21 & 0 & 0 \\ 4 & 24 & 0 \\ 0 & 4 & 40 \\ 0 & 0 & 4 \end{pmatrix}. \quad (5.29)$$

The world reference frame is located at the fixed base module frame. The joint inputs are $\{\theta_1, \theta_2, \theta_3\} = \{0, -\pi/6, 0\}$. The robot is shown in Fig. 5.7(b). ■

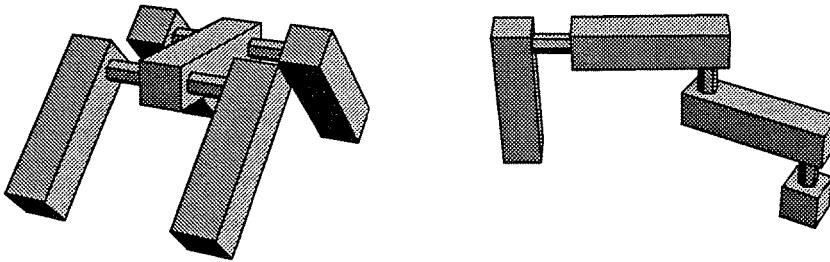


Figure 5.7: Forward kinematics examples

5.2. Kinematic Equivalence

As mentioned in the last chapter, equivalent AIMs have identical robot kinematics. Ignoring all labels on the modules and port numbers, these AIMs represent identical

robot constructions. However, the converse is not true. Because of the standardization on module component design and finite number of module assemblies, there exists a small subset of distinct AIMs which possess identical kinematic properties such as the shape and the size of the workspace and singularities.

Consider two distinct eAIMs of a fixed-base 3-DOF modular robot shown in Fig. 5.8. The robot has three identical prism modules and three R-joint modules. The eAIMs of the two assembly configurations are

$$A_1 = \begin{pmatrix} 1 & 0 & 0 & FB \\ 1 & 2 & 0 & L \\ 0 & 1 & 2 & L \\ 0 & 0 & 1 & L \\ R & R & R & 0 \end{pmatrix} \quad A_2 = \begin{pmatrix} 1 & 0 & 0 & FB \\ 1 & 6 & 0 & L \\ 0 & 1 & 2 & L \\ 0 & 0 & 1 & L \\ R & R & R & 0 \end{pmatrix}. \quad (5.30)$$

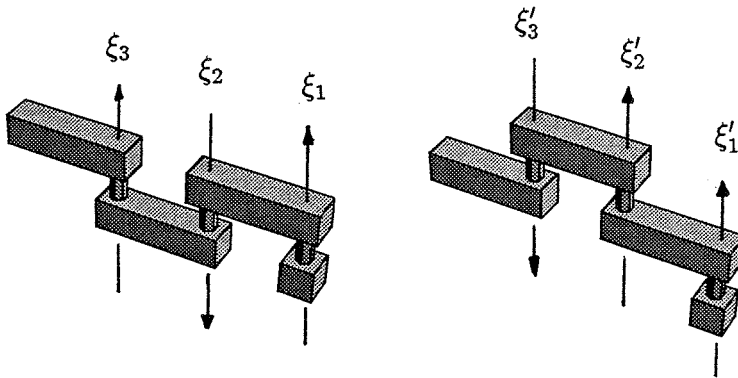


Figure 5.8: Kinematically equivalent robots

In Fig. 5.8, the robot base is drawn as a cube. According to Definition 4.12, A_1 and A_2 are inequivalent. Assembly configuration A_1 can be characterized by the twist coordinate of its joint axes, (ξ_1, ξ_2, ξ_3) . Similarly, A_2 is characterized by (ξ'_1, ξ'_2, ξ'_3) . At the initial positions, $\xi_1 = \xi'_1$, $\xi_2 = -\xi'_2$, and $\xi_3 = -\xi'_3$. However, ξ_2 and ξ'_2 , ξ_3 and ξ'_3 respectively represent identical joint axes with their directions reversed. From a kinematic performance point of view, these two assembly configurations can perform the same tasks.

Here we will establish a kinematic equivalence relation between two assembly configurations based on their joint twists. Because an AIM merely provides an algebraic

description of the robot assembly, the joint twists can be identified only after a kinematic model is imposed on the AIM.

Equivalence of 6R Manipulators

The equivalence of 6R robot kinematics using an intrinsic property, i.e., the joint axes, was first studied by Paden and Sastry [75]. The joint axes of a 6R manipulator are all assigned zero-pitched unit amplitude twist coordinate ξ_i , $i = 1, \dots, 6$. Then the ordered set of twists (from base to the end-effector), $\bar{\xi} = (\xi_1, \dots, \xi_6)$, is called a representative of the manipulator. Because a manipulator may have many different representatives corresponding to different “zero” postures and different senses of positive rotation of the joint axes, an equivalence relation is defined on the representatives of the same manipulator.

Definition 5.11: [75] Two representatives $\bar{\zeta}$ and $\bar{\xi}$ are equivalent, i.e., $\bar{\zeta} \sim \bar{\xi}$, if there exist joint offsets, $\bar{\phi} \in \mathbb{T}^5$, such that

$$\begin{aligned}\hat{\xi}_1 &= \pm \hat{\zeta}_1 \\ \hat{\xi}_i &= \pm (e^{\phi_1 \hat{\zeta}_1} e^{\phi_2 \hat{\zeta}_2} \dots e^{\phi_{i-1} \hat{\zeta}_{i-1}}) \hat{\zeta}_i (e^{\phi_1 \hat{\zeta}_1} e^{\phi_2 \hat{\zeta}_2} \dots e^{\phi_{i-1} \hat{\zeta}_{i-1}})^{-1}, \quad i = 2, \dots, 5\end{aligned}\quad (5.31)$$

This gives the physical interpretation of “ \sim ” as $\bar{\zeta} \sim \bar{\xi}$ if the axis of each ζ_i can be rotated successively about the previous axes $\zeta_{i-1}, \dots, \zeta_1$ such that its axis is coincident with that of ξ_i .

5.2.1. Equivalence of R-joint Serial Modular Robots

From now on, we restrict our attention to the kinematic equivalence of the regional structure of fixed-base serially connected R-joint modular robots. Once this is solved, it can be extended for a more general tree-structure modular robot. The twist coordinates of all joint axes of a serial modular robot can be obtained from an AIM using the `TreeRobotKinematics` algorithm along with a set of joint displacements. Thus, Definition 5.11 can be directly applied to determine the equivalence of modular robot joint twists. However, the AIM considered here is the regional structure of a robot; the

position/orientation of the end link affects the position/orientation of the end-effector attached to it. The geometric symmetry of the end-link allows a non-unique representation of the end-link orientation relative to the world reference frame. An extension of Definition 5.11 by considering the symmetric rotations of the end-link is proposed here for kinematic equivalence of AIMs.

Let $\bar{\xi}$ be the ordered set of joint twists obtained from an AIM of a k -DOF serial modular robot along with a set of joint angles $\bar{\theta}$. We write

$$\bar{\xi} \equiv (A, \bar{\theta}). \quad (5.32)$$

The forward kinematic map of the modular robot indicating the location of the end-link with respect to the base frame is then

$$f_{\bar{\xi}}(\theta_1, \dots, \theta_k) = e^{\theta_1 \hat{\xi}_1} \dots e^{\theta_k \hat{\xi}_k} T_{bk}(0), \quad (5.33)$$

where $T_{bk}(0)$ is the initial position/orientation of the end-link frame.

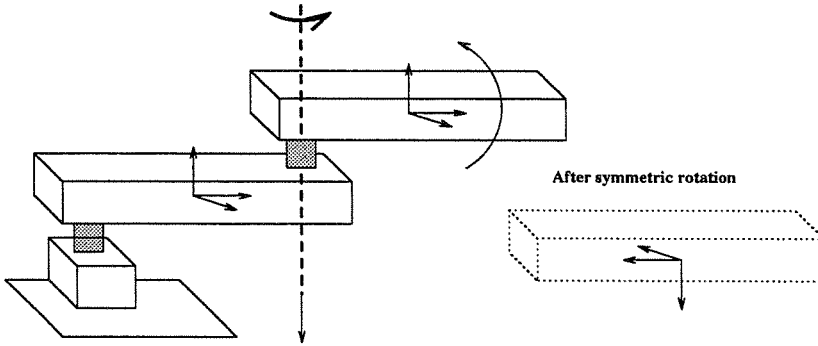


Figure 5.9: Symmetric rotation on the end link

Because of geometric symmetry, the last joint can be attached to a different port of the end-link without changing the physical appearance and function of the robot. This means that the end-link still occupies the same space but in a different orientation by a symmetry rotation as show in Fig. 5.9. An *extended* forward kinematic map, $f_{\bar{\xi}}^*$, that includes the symmetric rotation of the end-link is defined as

$$f_{\bar{\xi}}^*(\theta_1, \dots, \theta_k, \theta_e \xi_e) = e^{\theta_1 \hat{\xi}_1} \dots e^{\theta_k \hat{\xi}_k} e^{\theta_e \hat{\xi}_e} T_{bk}(0), \quad (5.34)$$

where ξ_e is the axis of symmetry rotation with respect to the world reference frame and θ_e is the angle of rotation. One can imagine that the symmetry rotation operation is performed by a virtual joint axis expressed by ξ_e after the rotation about the last joint axis, ξ_k .

Note that $e^{\theta_e \hat{\xi}_e} = \begin{pmatrix} R_e & \mathbf{p}_e \\ 0 & 1 \end{pmatrix}$ is a 4×4 homogeneous matrix corresponding to a symmetry rotation about the origin of the end-link frame. The rotation matrix, R_e , belongs to the symmetry rotation group of the end-link. The displacement \mathbf{p}_e is the location of the origin of the end-link frame relative to the world frame.

Definition 5.12: (*Kinematic Equivalence*)

Let $\bar{\xi}$ and $\bar{\zeta}$ be the twist sets obtained from AIMs A_1 and A_2 of a k -DOF robot along with joint angles $\bar{\theta}_1$ and $\bar{\theta}_2$, i.e., $\bar{\xi} = (A_1, \bar{\theta}_1)$ and $\bar{\zeta} = (A_2, \bar{\theta}_2)$. A_1 and A_2 are kinematically equivalent if and only if the following conditions are satisfied

1. $\bar{\xi} \sim \bar{\zeta}$, by Definition 5.11.
2. There exists a set of joint offsets, $\bar{\phi} \in \mathbb{T}^k$, and a symmetry rotation of the end-link, $\phi_e \zeta_e$, such that

$$f_{\bar{\xi}}^*(\theta_1, \dots, \theta_k, 0) = f_{\bar{\zeta}}^*(\pm\theta_1 + \phi_1, \dots, \pm\theta_k + \phi_k, \phi_e \zeta_e), \quad (5.35)$$

where ζ_e represents the twist coordinate of the symmetric rotation.

Example 5.13: Consider two planar 2-revolute-joint robots shown in Fig. 5.10. Their bases are coincident. The twist representation of a robot is given directly. Robot A and B are represented by $\bar{\xi} = \{\xi_1, \xi_2\}$ and $\bar{\zeta} = \{\zeta_1, \zeta_2\}$ respectively. The initial position/orientation of the end-link of robot A and B are $T_A(0)$ and $T_B(0)$ respectively. We have

$$f_{\bar{\xi}}^*(0, 0, 0) = T_A(0) \quad (5.36)$$

$$f_{\bar{\zeta}}^*(\phi_1, \phi_2, \phi_e \zeta_e) = e^{\phi_1 \hat{\zeta}_1} e^{\phi_2 \hat{\zeta}_2} e^{\phi_e \hat{\zeta}_e} T_B(0). \quad (5.37)$$

$\bar{\xi}$ is equivalent to $\bar{\zeta}$, because $\hat{\xi}_1 = \hat{\zeta}_1$ and $\hat{\xi}_2 = e^{\phi_1 \hat{\zeta}_1} \hat{\zeta}_2 e^{-\phi_1 \hat{\zeta}_1}$, where the offset angle, $\phi_1 = -\pi/2$. If the second link of robot B is rotated about the z-axis, ξ_z , of its own

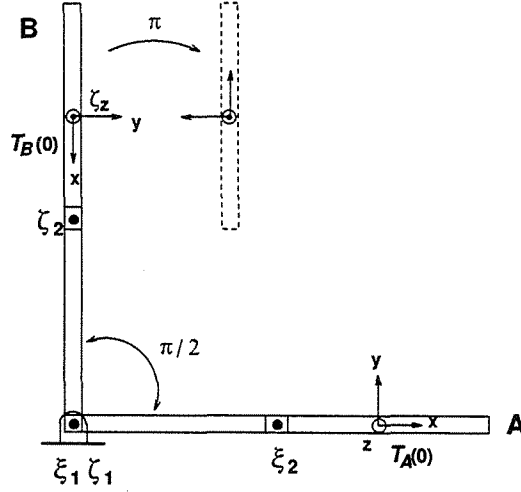


Figure 5.10: Two kinematically equivalent planar robots

module frame by angle π and then the entire robot is rotated about the ζ_1 axis by angle $\pi/2$, the end link of robot B will be coincident with that of robot A at its initial position, i.e.,

$$f_{\bar{\xi}}^*(0, 0, 0) = f_{\bar{\zeta}}^*(\phi_1, \phi_2, \phi_e \zeta_e), \quad (5.38)$$

where the offset angles, $\phi_2 = 0$, $\phi_e = \pi$, and $\zeta_e = \zeta_z$. The end-links of both robots will generate identical workspace and joint singularities. From (5.38), we have

$$T_B(0) = e^{-\phi_e \hat{\zeta}_e} e^{-\phi_2 \hat{\zeta}_2} e^{-\phi_1 \hat{\zeta}_1} T_A(0). \quad (5.39)$$

For other joint angles,

$$\begin{aligned} f_{\bar{\xi}}^*(\theta_1, \theta_2, 0) &= e^{\theta_1 \hat{\xi}_1} e^{\theta_2 \hat{\xi}_2} T_A(0) \\ &= e^{\pm \theta_1 \hat{\zeta}_1} e^{\pm \theta_2 (e^{\phi_1 \hat{\zeta}_1} \hat{\zeta}_2 e^{-\phi_1 \hat{\zeta}_1})} T_A(0) \\ &= e^{\pm \theta_1 \hat{\zeta}_1} (e^{\phi_1 \hat{\zeta}_1} e^{\pm \theta_2 \hat{\zeta}_2} e^{-\phi_1 \hat{\zeta}_1}) T_A(0) \\ &= e^{(\pm \theta_1 + \phi_1) \hat{\zeta}_1} e^{\pm \theta_2 \hat{\zeta}_2} e^{-\phi_1 \hat{\zeta}_1} T_A(0) \\ &= e^{(\pm \theta_1 + \phi_1) \hat{\zeta}_1} e^{\pm \theta_2 \hat{\zeta}_2} e^{\phi_2 \hat{\zeta}_2} e^{\phi_e \hat{\zeta}_e} e^{-\phi_e \hat{\zeta}_e} e^{-\phi_2 \hat{\zeta}_2} e^{-\phi_1 \hat{\zeta}_1} T_A(0) \\ &= e^{(\pm \theta_1 + \phi_1) \hat{\zeta}_1} e^{(\pm \theta_2 + \phi_2) \hat{\zeta}_2} e^{\phi_e \hat{\zeta}_e} T_B(0) \\ &= f_{\bar{\zeta}}^*(\pm \theta_1 + \phi_1, \pm \theta_2 + \phi_2, \phi_e \zeta_e) \end{aligned}$$

So robot A and B are kinematically equivalent. ■

5.2.2. Equivalence Test Procedure

From a computational point of view, the implementation of the recursive definition of the equivalence of manipulator twists in (5.31) is time consuming. The equivalence of ξ_i and ζ_i involves two pairs of successive joint twists, ξ_{i-1}, ξ_i and ζ_{i-1}, ζ_i . The relative orientation between two twists of revolute joints can be characterized by two parameters: d , the shortest distance between two joint axes, and α , the skew angle between the direction of the joint axes as shown in Fig. 5.11.

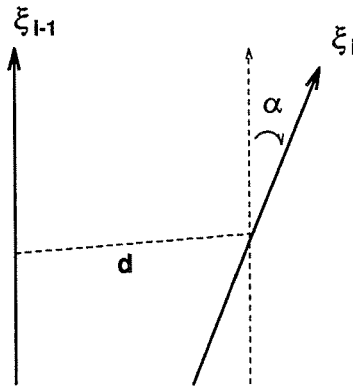


Figure 5.11: Joint axes parameters

Let $\xi_{i-1} = (\mathbf{v}_{i-1}, \mathbf{w}_{i-1})$ and $\xi_i = (\mathbf{v}_i, \mathbf{w}_i)$; the skew angle and the shortest distance between ξ_{i-1} and ξ_i can be obtained by

$$\alpha = \tan^{-1} \frac{\|\mathbf{w}_{i-1} \times \mathbf{w}_i\|}{\mathbf{w}_{i-1} \cdot \mathbf{w}_i} \quad (5.40)$$

$$d = -\frac{\mathbf{v}_{i-1} \cdot \mathbf{w}_i + \mathbf{v}_i \cdot \mathbf{w}_{i-1}}{\sin \alpha}. \quad (5.41)$$

A pair of successive twists can be characterized by the pair (d, α) . Two pairs of twists are considered to have identical relative orientation if $d_1 = d_2$ and $\alpha_1 = \alpha_2$. However, the twists may point in the opposite direction of the line. Two pairs of successive twists are still considered having identical relative orientation if the skew angles differ by π , i.e., $|\alpha_1 - \alpha_2| = \pi$. If the relative orientation cannot match, ξ_i and ζ_i are definitely inequivalent.

If the relative orientations are identical, the offset angle ϕ_i can be found as follows. Let $\zeta_{i-1} = (\mathbf{v}_{i-1}^*, \mathbf{w}_{i-1}^*)$ and $\zeta_i = (\mathbf{v}_i^*, \mathbf{w}_i^*)$. For parallel joint axes, let $u' = \mathbf{w}_i^* - (\mathbf{w}_{i-1}^* \mathbf{w}_{i-1}^{*T}) \mathbf{w}_i^*$ and $v' = \mathbf{w}_i - (\mathbf{w}_{i-1}^* \mathbf{w}_{i-1}^{*T}) \mathbf{w}_i$, then

$$\phi_i = \tan^{-1} \frac{\mathbf{w}_{i-1}^* \cdot (u' \times v')}{u' \cdot v'}. \quad (5.42)$$

For non-parallel joint axes, we have

$$\mathbf{w}_i^* \times (\mathbf{v}_i - \mathbf{v}_i^*) = \tan \frac{\phi_i}{2} (2\mathbf{v}_{i-1}^* - (\mathbf{w}_{i-1}^* \cdot \mathbf{w}_i^*)(\mathbf{v}_i + \mathbf{v}_i^*)). \quad (5.43)$$

Substitute ϕ_i back to equation (5.31); the equivalence of successive twists can be determined. The procedure to check the equivalence of two twists ξ_i and ζ_i can be summarized as follows.

1. Start with two pairs of twists, ξ_{i-1} , ξ_i , and ζ_{i-1} , ζ_i .
2. Compute the pairs (d_1, α_1) for ξ_{i-1} and ξ_i , and (d_2, α_2) for ζ_{i-1} and ζ_i from equations (5.40) and (5.41).
3. If $d_1 = d_2$ and $\alpha_1 = \alpha_2$, or $d_1 = d_2$ and $|\alpha_1 - \alpha_2| = \pi$, then
 - (a) Find the offset angle ϕ_i from (5.42) and (5.43).
 - (b) Substitute ϕ_i into (5.35) to check the equivalence of ξ_i and ζ_i .

If not, then ξ_i and ζ_i are inequivalent.

Example 5.14: Consider the 3-DOF fixed base robots in Example 4.18. There are 98 distinct AIMs, but by Definition 5.12, there are two pairs of AIMs which are kinematically equivalent as shown in Fig. 5.12 and 5.13.

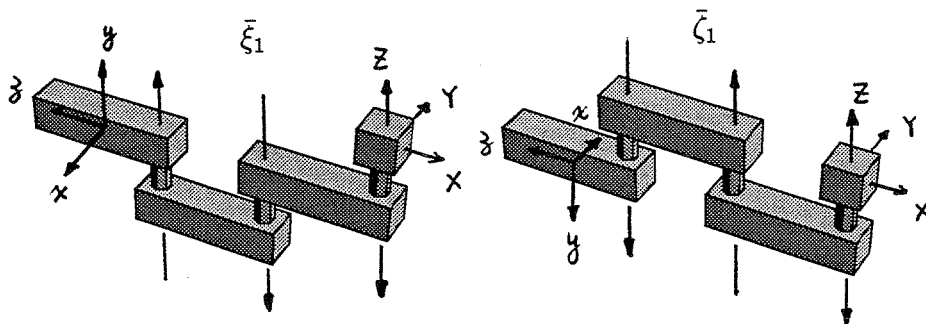


Figure 5.12: 3-DOF kinematically equivalent robots (A)

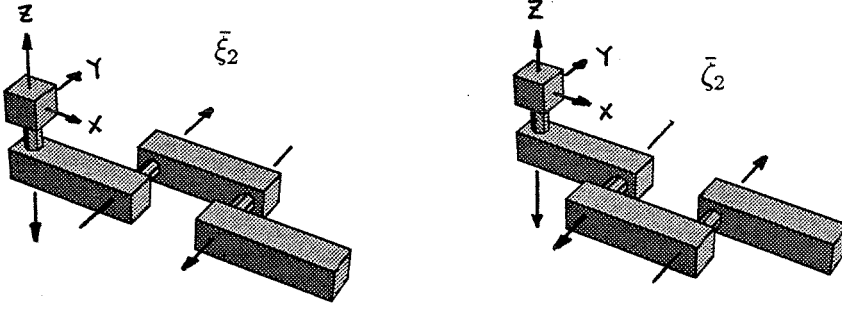


Figure 5.13: 3-DOF kinematically equivalent robots (B)

The AIMs of Pair (A) are

$$A_{11} = \begin{pmatrix} 5 & 0 & 0 & FB \\ 4 & 7 & 0 & L \\ 0 & 7 & 8 & L \\ 0 & 0 & 8 & L \\ R & R & R & 0 \end{pmatrix} \quad A_{12} = \begin{pmatrix} 5 & 0 & 0 & FB \\ 7 & 8 & 0 & L \\ 0 & 7 & 8 & L \\ 0 & 0 & 8 & L \\ R & R & R & 0 \end{pmatrix} \quad (5.44)$$

The joint twists are

$$\bar{\xi}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -3 & 6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & -1 & 1 \end{pmatrix}, \quad \bar{\zeta}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3 & -6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 1 & -1 \end{pmatrix}. \quad (5.45)$$

Therefore, $\xi_1 = \zeta_1$, $\xi_2 = -\zeta_2$, and $\xi_3 = -\zeta_3$. The offset angles, $\phi_1 = \phi_2 = 0$. The position of the end links are

$$T_{\bar{\xi}_1} = \begin{pmatrix} 0 & 0 & -1 & -7.5 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad T_{\bar{\zeta}_1} = \begin{pmatrix} 0 & 0 & -1 & -7.5 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (5.46)$$

Let the z-axis of the end link frame be the symmetric rotation's axis. The twist of this axis is $\xi_e = (0, 2, 0, -1, 0, 0)^T$. The symmetric rotation about ξ_e by angle π becomes

$$e^{\pi \hat{\xi}_e} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -4 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (5.47)$$

We have

$$f_{\bar{\xi}_1}^*(0, 0, \pi \xi_e) = e^{\pi \hat{\xi}_e} T_{\bar{\xi}_1} = T_{\bar{\zeta}_1} = f_{\bar{\zeta}_1}^*(0, 0, 0). \quad (5.48)$$

Similarly, the AIMs of Pair (B) are

$$A_{21} = \begin{pmatrix} 5 & 0 & 0 & FB \\ 5 & 8 & 0 & L \\ 0 & 7 & 8 & L \\ 0 & 0 & 8 & L \\ R & R & R & 0 \end{pmatrix} \quad A_{22} = \begin{pmatrix} 5 & 0 & 0 & FB \\ 6 & 7 & 0 & L \\ 0 & 7 & 8 & L \\ 0 & 0 & 8 & L \\ R & R & R & 0 \end{pmatrix}. \quad (5.49)$$

The joint twists are

$$\bar{\xi}_2 = \begin{pmatrix} 0 & 2 & -2 \\ 0 & 0 & 0 \\ 0 & 3 & -6 \\ 0 & 0 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 0 \end{pmatrix}, \quad \bar{\zeta}_2 = \begin{pmatrix} 0 & -2 & 2 \\ 0 & 0 & 0 \\ 0 & -3 & 6 \\ 0 & 0 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & 0 \end{pmatrix}. \quad (5.50)$$

Therefore, $\xi_1 = \zeta_1$, $\xi_2 = -\zeta_2$, and $\xi_3 = -\zeta_3$. The offset angles, $\phi_1 = \phi_2 = 0$. The position of the end links are

$$T_{\bar{\xi}_2} = \begin{pmatrix} 0 & 0 & 1 & 7.5 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad T_{\bar{\zeta}_2} = \begin{pmatrix} 0 & 0 & 1 & 7.5 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (5.51)$$

Let the z-axis of the end link frame be the symmetric rotation's axis. The twist of this axis is $\xi_e = (0, -2, 0, 1, 0, 0)^T$. The symmetric rotation about ξ_e by angle π is identical to (5.47). We have

$$f_{\bar{\xi}_2}^*(0, 0, \pi\xi_e) = e^{\pi\hat{\xi}_e} T_{\bar{\xi}_2} = T_{\bar{\zeta}_2} = f_{\bar{\zeta}_2}^*(0, 0, 0). \quad (5.52)$$

So A_{11} is kinematically equivalent to A_{12} and A_{21} is kinematically equivalent to A_{22} . ■

5.3. Inverse Kinematics

The robot inverse kinematics problem is concerned with finding the joint angles that cause a mechanism to reach a desired position/orientation of the end-effector. This is especially important when we want to control the robot motion or to evaluate the kinematic performance at a task point.

To find a closed form solution to the inverse kinematics problem of a general serial manipulator attracted the attention of many researchers. The inverse kinematics of a general 6R robot contain a set of highly nonlinear trigonometric equations in terms of

the joint angles. By using the substitution for every joint angle θ_i ,

$$u_i = \tan \frac{\theta_i}{2}. \quad (5.53)$$

For $\sin \theta_i$ and $\cos \theta_i$, one obtains

$$\sin \theta_i = \frac{2u_i}{1 + u_i^2} \quad \text{and} \quad \cos \theta_i = \frac{1 - u_i^2}{1 + u_i^2}. \quad (5.54)$$

The trigonometric equations are transformed into a set of multivariate polynomial equations, which can be solved by numerical continuation technique proposed by Wampler, Morgan and Sommese [97], Tsai and Morgan [92,92], or by elimination method proposed by Raghavan and Roth [84,85].

These methods can deal with manipulators with five or six revolute joint whose joint axes are in either general or special geometries such as intersecting axes. However, in a modular robot, there is no fixed assembly configuration; hence, the geometry of joint axes varies. Furthermore, the number of a robot's degree of freedom can be changed. Thus, it is generally not possible to find a close form solution for inverse kinematics.

In order to provide for generality, we employ a numerical inverse kinematics (NIK) scheme proposed by Khosla, Neuman, and Prinz [47] to solve the inverse kinematics of modular robots. This algorithm is also used in solving the inverse kinematics of RMMS [46]. The block diagram of this scheme is shown in Fig. 5.14.

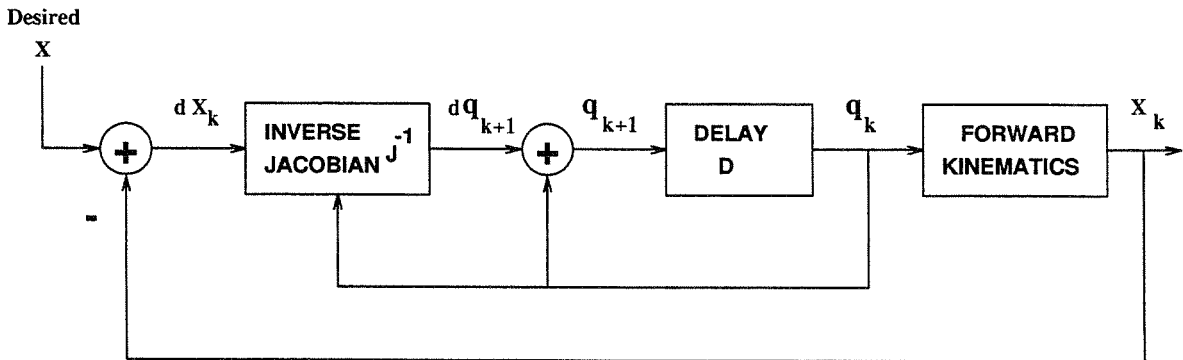


Figure 5.14: Block diagram of the NIK (after [47])

This is a close-loop scheme using Newton-Ralphson iteration. The iteration determines the necessary change in the joint angles to achieve a differential change in the position/orientation of the end-effector. The forward kinematics is described by

$$\mathbf{x} = f(\mathbf{q}), \quad (5.55)$$

where \mathbf{x} is the Cartesian position and orientation (in Euler angles usually) of the end-effector and \mathbf{q} is a vector of joint angles. The manipulator Jacobian relates the differential change $d\mathbf{x}$ and $d\mathbf{q}$:

$$d\mathbf{x} = J(\mathbf{q}) d\mathbf{q}. \quad (5.56)$$

For non-redundant manipulators, the change in $d\mathbf{q}$ can be written as

$$d\mathbf{q} = J^{-1}(\mathbf{q}) d\mathbf{x}. \quad (5.57)$$

This equation can be written in an iteration form as

$$d\mathbf{q}_{k+1} = J^{-1}(\mathbf{q}_k) d\mathbf{x}_k, \quad (5.58)$$

where k is the number of iteration. Then the joint displacement can be updated as

$$\mathbf{q}_{k+1} = \mathbf{q}_k + d\mathbf{q}_{k+1}. \quad (5.59)$$

We then solve equation (5.58) and (5.59) iteratively until x_k is within the required error tolerance ϵ of the desired \mathbf{x}_d , i.e., $|\mathbf{x}_d - \mathbf{x}_k| < \epsilon$.

This NIK scheme will be employed in solving inverse kinematics of a modular robot for the following two conditions: (1) for serial type modular robot only; (2) the inverse kinematics solution will be a set of joint angles that achieves a desired position in the end link, which is described by the location of the origin of the end link frame.

The technique to solve the inverse kinematics of a serial robot can be also applied to robots with star-like topologies, such as a multifinger robot hand where each finger is considered as a serial manipulator. Because each finger can be actuated independently, a hand Jacobian can be formulated by stacking the Jacobian of all fingers in block diagonal form.

Because the kinematics and reach of a modular robot regional structure is the major concern here, it is feasible to find the inverse kinematics solution for a given end link position. Under these circumstances, the robot Jacobian is a $3 \times n$ matrix, termed the end link Jacobian J^e , which relates the change in joint angles and the Cartesian location of the origin of the end link frame. n is the number of DOF in the regional structure. Substituting J^e to equations (5.56) to (5.58), the NIK of a modular robot can be established. Note that J^e is very different from the spatial Jacobian composed by the twist coordinate of the joint axes defined in (5.26). The derivation of J^e is described in Section 5.3.1.

A regional structure with 3 DOF is a non-redundant modular robot since the Jacobian is an invertible 3×3 matrix. For a regional structure with DOF more than 3, the robot introduces redundancy which will complicate the computation of inverse kinematics solution. The Jacobian J^e is no longer invertible and a generalized inverse must be provided for the inverse operation in (5.58).

A singularity robust inverse, J^* , is a generalized inverse suitable for this operation and is employed in RMMS [46]. It is given by [71]

$$J^* = J^T (J^T J + \lambda I)^{-1}, \quad (5.60)$$

where λ is the scale factor which will be adjusted automatically according to the manipulator's distance from a singular point. This singularity robust inverse will generate feasible solutions in the neighborhood of singular points.

Note that by the non-uniqueness of the solution of inverse kinematics, the solution of NIK is initial condition dependent, i.e., given different initial joint angles and the position of the end effector, one may find different solution to the same desired task point. In most of the application, one solution is enough. However, choosing random initial robot postures may provide us with a set of inverse kinematics solutions.

5.3.1. Derivation of the End Link Jacobian

The end link Jacobian, J^e , is defined to be the matrix relating the changes in joint angles with the changes in the Cartesian coordinate of the origin of the end link frame. Let $g_{we} \in SE(3)$ be the position/orientation of the end link frame relative to the world reference frame, where

$$g_{we} = \begin{pmatrix} R_{we} & \mathbf{p}_{we} \\ 0 & 1 \end{pmatrix}. \quad (5.61)$$

Let J^s be the spatial Jacobian composed of the joint twists written in the world frame defined by (5.26) and J^b , the body Jacobian written in the end link frame. They are related by [69]

$$J^s = \text{Ad}_{g_{we}} J^b. \quad (5.62)$$

Both J^s and J^b are $6 \times n$ matrices, where n is the DOF of the regional structure of the robot. The general velocity of the end link written in its own module (body) frame is given by

$$V_e^b = J^b \dot{\theta} = \begin{pmatrix} \mathbf{v}_e^b \\ \mathbf{w}_e^b \end{pmatrix}, \quad (5.63)$$

where \mathbf{v}_e^b is the velocity of the origin of the end link frame relative to the world frame as viewed in the current module frame, and \mathbf{w}_e^b is the angular velocity of the end link frame as viewed in the current module frame. Now assume a coordinate frame w' whose origin is coincident with the end link frame, e , and axes are parallel to the world frame, w , is defined as shown in Fig. 5.15.

The linear velocity \mathbf{v}_e^b and the angular velocity \mathbf{w}_e^b relative to frame w' is thus

$$v_{w'}^b = \begin{pmatrix} R_{w'e} & 0 \\ 0 & R_{w'e} \end{pmatrix} V_e^b \equiv \mathbb{R}_{w'e} V_e^b. \quad (5.64)$$

Note that $v_{w'}^b$ is a composite velocity describing both the linear and angular velocity of frame e . Because frame w' is parallel to the world frame w , $R_{ww'} = I$ and $R_{we} = R_{ww'} R_{w'e} = R_{w'e}$. The composite velocity relative to w is given by

$$v_w^b = \begin{pmatrix} R_{ww'} & 0 \\ 0 & R_{ww'} \end{pmatrix} v_{w'}^b = \mathbb{R}_{we} V_e^b. \quad (5.65)$$

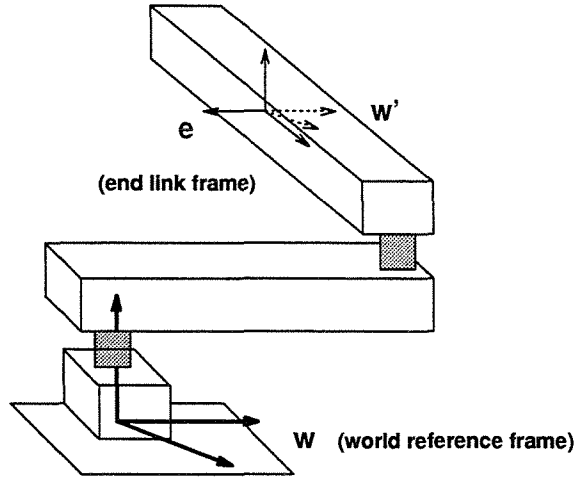


Figure 5.15: Relations among frames w , w' , and e

Substituting (5.63) into (5.65) and from (5.62) we have

$$\mathbf{v}_w^b = \mathbb{R}_{we} J^b \dot{\theta} = \mathbb{R}_{we} \text{Ad}_{g_{we}}^{-1} J^s \dot{\theta}. \quad (5.66)$$

Because

$$\text{Ad}_{g_{we}} = \begin{pmatrix} R_{we} & S(\mathbf{p}_{we}) R_{we} \\ 0 & R_{we} \end{pmatrix}, \quad (5.67)$$

$$\text{Ad}_{g_{we}}^{-1} = \begin{pmatrix} R_{we}^T & -R_{we}^T S(\mathbf{p}_{we}) \\ 0 & R_{we}^T \end{pmatrix}. \quad (5.68)$$

Hence,

$$\begin{aligned} \mathbf{v}_w^b &= \begin{pmatrix} R_{we} & 0 \\ 0 & R_{we} \end{pmatrix} \begin{pmatrix} R_{we}^T & -R_{we}^T S(\mathbf{p}_{we}) \\ 0 & R_{we}^T \end{pmatrix} J^s \dot{\theta} \\ &= \begin{pmatrix} I & -S(\mathbf{p}_{we}) \\ 0 & I \end{pmatrix} J^s \dot{\theta} \\ &= J^E \dot{\theta}. \end{aligned} \quad (5.69)$$

Note that J^E is a $6 \times n$ matrix. Since only the velocity of the origin of the end link frame, \mathbf{v}_w^b , is concerned, denoting J^E by two $3 \times n$ submatrices, J_1^E and J_2^E , we have

$$\mathbf{v}_w^b = \begin{pmatrix} \mathbf{v}_w^b \\ \mathbf{w}_w^b \end{pmatrix} = \begin{pmatrix} J_1^E \\ J_2^E \end{pmatrix} \dot{\theta}. \quad (5.70)$$

Finally,

$$\mathbf{v}_w^b = J_1^E \dot{\theta}, \quad (5.71)$$

where J_1^E defines the end link Jacobian, J^e , i.e., $J_1^E = J^e$.

5.4. Discussion

This chapter formulated the modular robot forward kinematics based on the Product-of-Exponential model in two stages. First, the local representation of a joint twist is introduced by applying port conversion functions. The forward transformation of a dyad is derived from a set of initial position functions defined according to the connection of the dyad. The tree-structured robot forward kinematics can be derived from this dyad kinematics along with a tree-traversing algorithm. Equipped with a kinematic model, the kinematic equivalence of AIMS can be defined. Kinematic equivalent AIMS have identical kinematic properties. Finally, a numerical inverse kinematics scheme based on Newton-Raphson method is introduced for solving the joint angles for a desired end link position. The next chapter will show that both the kinematic equivalence of AIMS and the inverse kinematics solutions are crucial to the task-optimal configuration problem.

Chapter 6

Task-Oriented Optimal Configurations

This chapter discusses issues pertaining to the optimal assembly configuration of a modular robot for a given task. Owing to the standardized module design and multiple connection methods on link modules, a modular robot can be reconfigured freely using the same set of modules. One can generate all possible assembly configurations and test each one against the task requirements to find the best one for a particular robot task. However, a reconfiguration may result in the change of robot topology which will, in turn, alter the basic function of the robot. As mentioned before, robots with different topologies are functionally different. Thus, how to define a robot task conformed with its associated topology and an objective criteria to evaluate the task performance of a robot become imperative subjects. Furthermore, the existence of multiple connections on a link module may create joint patterns with undesirable kinematics, such as joint redundancy and link interference. From a task point of view, these issues affect the performance of a modular robot and should be considered.

Paredis and Khosla [76] have considered task-based robot design for fixed configuration robots. In their work, tasks are defined as a set of working points in the task space. A *kinematic space*, which is the Cartesian product of D-H configuration space of links, the joint space, and the task space, is introduced. A manipulator in a certain posture at a certain working point can be represented as a point in the kinematic space.

By formulating robot inverse kinematics in a closed form, all task specifications can be transformed into equalities and inequalities in the kinematic space. The design of a manipulator becomes an optimization problem in the kinematic space, where the objective function is defined according to the task specification. Because all design parameters are continuous, conventional optimization techniques, such as gradient descent methods, can be employed. However, in a modular reconfigurable robot, all of the design parameters, such as link lengths and connecting port locations, are pre-determined in the module level. The kinematic space that corresponds to all assembly configurations is a discrete set, so these continuous optimization techniques cannot apply.

Here the task-oriented optimal assembly configuration problem is formulated as an optimization problem by defining a task related objective function. This function evaluates a modular robot assembly configuration for a given task while maintaining desirable joint patterns in the robot construction. The descriptions of a task is taken as parameters to the objective function. The search space is the set of assembly configurations, a discrete set instead of a continuous parameter set. A combinatorial optimization technique termed *genetic algorithms* is employed for this problem because of the discrete nature of the assembly configuration set.

This chapter is organized as follows. Section 6.1 discusses the general framework in solving this task-optimal configuration problem. Section 6.2 and 6.3 discuss the task and structure specifications required in a task description respectively. Section 6.4 formulates the optimization function of serial type modular robots based on the task and structure specifications. Section 6.5 introduces the genetic algorithm and the coding scheme for AIMS employed in this optimization problem. Examples that demonstrate this problem solving strategy are given in Section 6.6.

6.1. General Framework

To develop a solution to the task-optimal configuration problem, we pose it as an optimization problem. A task-oriented optimization function, termed an *assembly*

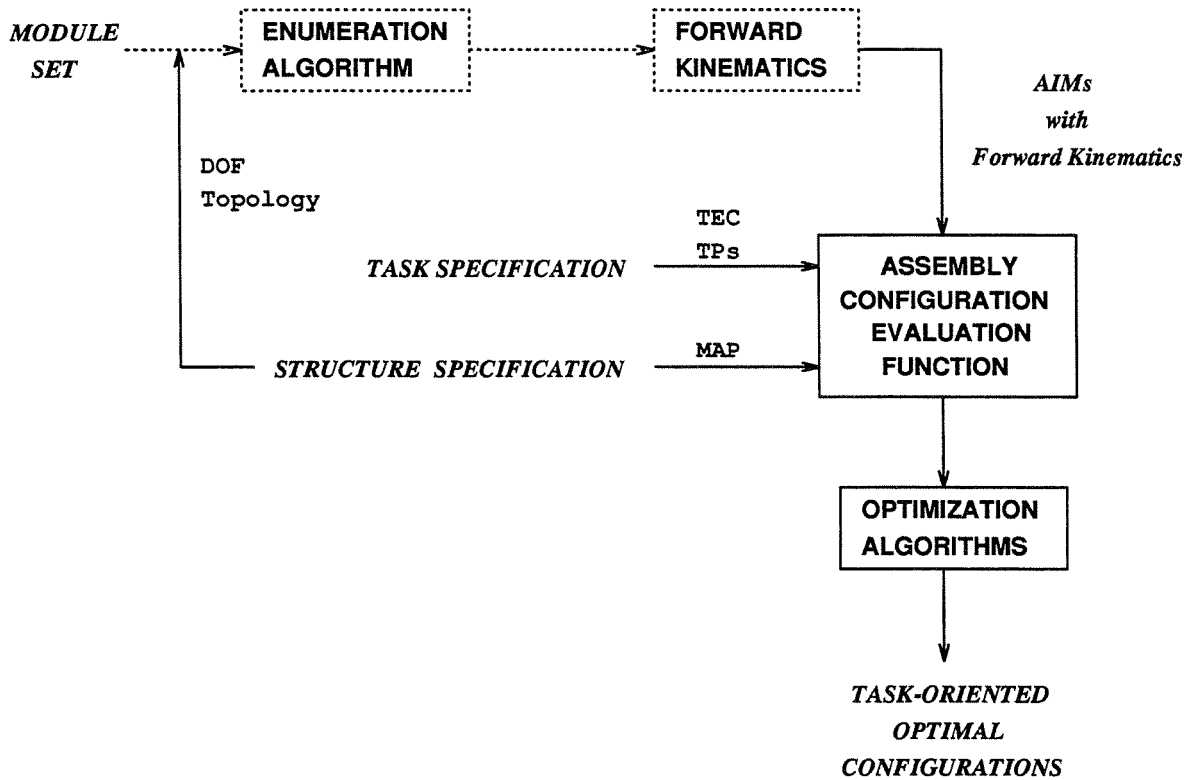


Figure 6.1: Framework for task-optimal configuration problem

configuration evaluation function (ACEF), which evaluates the task performance of an AIM, is formulated. The description of a robot task becomes parameters of the ACEF. The general framework solving this problem is shown in Fig. 6.1.

Because of the high complexity in finding a robot assembly which is not only task-optimal but also satisfies kinematic constraints, the task requirement is divided into task and structure specifications. Task specifications directly define and evaluate a robot task, and structure specifications contain the kinematic constraints that must be satisfied by the robot while executing the task.

The task specification gives a robot task at executable level and an objective criteria to judge the robot performance. An executable robot task means a sequence of motions and positions of the robot mechanism. Using this task definition, one can immediately decide the capability of the robot to achieve the task. A task evaluation criterion (TEC)

is necessary for judging the performance of different robot configurations executing an identical task. The criterion should also reflect the objective of a task requirement. For example, the TEC for tasks emphasizing positioning accuracy will be very different from that emphasizing force transmission ratio even if the sequence of motion and positions of the tasks are identical.

The structure specification describes the robot topology, the number of robot degree-of-freedom (DOF), and a function termed the *module assembly preference* (MAP). Robot topology is directly related to the function of the robot mechanism. It must match the class of the robot task given in the task specification. Different numbers of DOFs and different robot topologies alter the kinematics and dynamics of the assembled robot, and there is no common ground to compare robot with different number of DOFs and topologies. Hence, this information is specified prior to the enumeration algorithm. As mentioned in Chapter 4, the proposed enumeration method can handle robots with predetermined robot topology, so the input to the ACEF is a set of AIMs with a predefined number of DOFs and robot topology.

In order to satisfy the kinematic constraint on the joint assembly patterns in an optimal robot configuration, an auxiliary function, module assembly preference (MAP), is defined on all of the distinct joint patterns. The MAP is a binary function whose value is either a zero or a one. A zero is assigned to an unwanted joint pattern; a one, to an acceptable pattern. Multiplying the MAPs of all joint patterns, a preference value of the entire AIM, which is also a zero or a one, can be obtained. A zero shows the AIM possesses inappropriate joint patterns.

The overall ACEF that evaluates a robot executing a task while rejecting undesirable kinematic constraints is defined to be the product of the TEC and the preference value of the robot. The optimization problem is then stated as:

| |
|---|
| GIVEN: a robot task, a TEC, a MAP, and prescribed robot topology |
|---|

| |
|---|
| FIND: an AIM in the assembly configuration set whose ACEF value achieves maximum |
|---|

Because the robot task is closely related to its topology, the actual form of ACEF varies. In the following sections, we focus on finding task-optimal assembly configurations of a serial type fixed base modular arm with R-joints to demonstrate this problem solving strategy.

6.2. Task Specifications

6.2.1. Definition of Robot Tasks

A robot task can be defined at an abstract and descriptive level such that only the motion of the robot and the environment it is interacted with are given. As an example, consider the robot task: “grasp a cup and move it from A to B.” From this kind of task description, a task planner would construct robot trajectories and motion sequences at the executable level for task execution.

The execution level motion sequence and robot trajectory are adopted as the definition of a robot task for two reasons. First, with a prescribed robot motion and trajectory, one can immediately determine the capability of a modular robot configuration to carry out the task. If the robot is unable to carry out the motion, it is unnecessary to proceed any further with the task evaluation procedure. Secondly, it is very difficult to define a measure to quantify the goodness of a robot task with a very vague definition. An objective measure can be defined on this task specification for evaluation.

In this chapter, a robot task is defined to be a collection of working points, w_p , in the operation space \mathbb{R}^3 [33,48] or a collection of the end-effector positions/orientations $w_p = (x, y, z, \theta, \phi, \psi)$, where θ , ϕ , and ψ are Euler angles representing the orientation of the end-effector frame. If the robot is to follow a trajectory, say a spray painting robot or an arc welding robot following a prescribed path, this task can be approximated by a set of points along the path. (Fig. 6.2) For more specific purposes, the force/moment at the end-effector and position accuracy at the working points can be also included

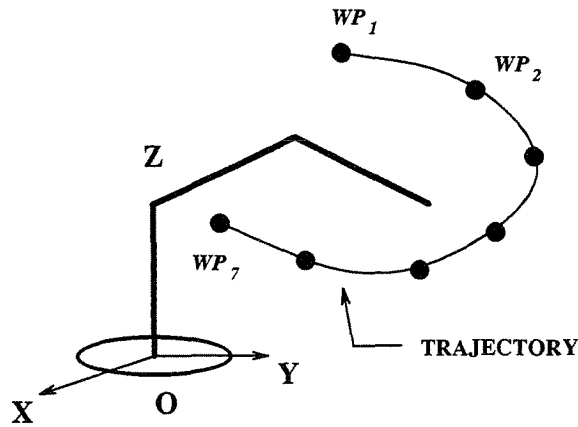


Figure 6.2: Definition of a Robot Task

as part of the definition of a task [33].

6.2.2. Task Evaluation Criteria

A task evaluation criteria (TEC) should be defined on a robot assembly configuration while taking the task description as the input parameter. Thus, one can use a TEC to compare the performance of different robot configurations executing an identical task. To achieve this goal, a TEC should satisfy the following requirements:

1. It should reflect the essence of the task definition. In the serial robot case, the TEC is defined on each working point because a single working point alone can be defined as a task. A method to extend it to a collection of working points is described in the following context.
2. It is a monotonic increasing positive real valued function so that an AIM with large TEC value represents a better configuration.
3. It should meet the requirement of the robot's task objective. For instance, a certain positioning accuracy or maximum force transmission ratio at the working points might be useful TECs.

Since a robot task is a set of discrete working points, many design criteria or kinematic performance measures of a robot manipulator can be employed for task evaluation [2,16,

50,51,107]. These performance measures evaluate the performance of a manipulator at a particular working point with a specified posture, and hence are called *local* measures.

These local measures often utilize the Jacobian matrix, J , of the manipulator structure. For example, the product of all singular values of a Jacobian, $\det(JJ^T)^{1/2}$, is called the *manipulability measure* by Yoshikawa [107,108]. This manipulability measure represents the volume of the manipulability ellipsoid in the configuration space of the manipulator. The condition number of a Jacobian, $\text{Cond}(J) = \sigma_{max}/\sigma_{min}$, is used by Klein and Blaho [50,51] to measure the dexterity of a manipulator posture. The condition number indicates the uniformity of the Jacobian transformation with respect to the direction of joint rates. The minimum singular value of the Jacobian, $\min \sigma(J)$, is also proposed by Klein and Blaho [50] as a measure of the closeness of the manipulator postures to singularities. The reciprocal of the condition number, termed the *conditioning index* (CI), has been proposed by Angeles [2] to measure the closeness of a configuration to a singularity. The range of CI is from 0 to 1. CI becomes zero if the manipulator is in singular position. A robot is called isotropic if its CI attains 1.

A local task-oriented performance measure has been explored by Chiu [16]. The tasks were defined to be a set of force and velocity vectors of interest. A *task compatibility index* based on the summation of the distances from the center to the boundary of the manipulability ellipsoid in the direction of interest is then defined.

The manipulability, the minimum singular value, and the conditioning index of a manipulator Jacobian satisfy all three requirements, therefore, any one of them can be chosen as a TEC for modular robots. The manipulability is a more favorable TEC because it is translational invariant properties [56]. That means its value is not altered as the base reference frame changes.

Let μ_i be the value of the TEC of a modular robot at a task point i . μ_i is obtained by substituting the solution of the inverse kinematics for task point i to the TEC. Distinct AIMS possess different robot kinematics; hence, the solution of the inverse kinematics for identical task differs. For a single task point, μ_i suffices to represent the performance

of a robot. However, for a collection of n task points, the total performance μ of the task is defined to be the smallest μ_i among the task points, i.e.,

$$\mu = \min_{i \in \{1, \dots, n\}} \mu_i . \quad (6.1)$$

μ represents the worst case among the collection of task points. Since $\mu_i \geq 0$, by definition, $\mu \geq 0$. Taking the minimum of μ_i as the total performance of a robot ensures that one robot assembly will be better than the other in the worst case.

Example 6.1: Let's consider using the 3-DOF serial robot of Example 5.10 to execute a task in the operation space. The manipulability is chosen to be the TEC. We require that the origin of the end link frame pass through three task points. The points and the corresponding manipulabilities are listed in Table 6.1. Since the task is a set of task points, by definition, the total performance μ is the smallest manipulability, i.e., $\mu = 16.0997$. ■

| | Task Point | Manipulability |
|---|----------------|----------------|
| 1 | (-3.5,3.5,3.) | 16.0997 |
| 2 | (-3.5,4.0,3.5) | 21.3534 |
| 3 | (-2.5,5.0,4.0) | 23.1894 |

Table 6.1: The manipulability of task points

6.3. Structure Specifications

6.3.1. DOF Selection

It is assumed that the number of DOFs required by a task will be provided prior to the assembly enumeration process for the generation of candidate assembly configurations. In general, six DOFs are necessary for spatial positioning and orienting. However, not all tasks require six DOFs. For example, when an IC is inserted to a printed circuit board (PCB), one DOF is required to orient the IC in the plane of the PCB and three DOFs are necessary to locate the position, so four DOFs are the minimum

requirement. When a tool at the end-effector has an orientational symmetry as in arc welding task, the minimum number of DOFs becomes five. On the other hand, when there are obstacles in the task space or the working environment is highly constrained [15] and more dexterity is required to perform the task, the minimum number of DOFs may be more than six.

In this thesis, we consider the kinematics of the regional structure of a modular robot without the end-effector (or wrist). A 3-DOF spherical wrist or 2-DOF wrist is considered as a separate module unit to be attached to the connecting port on the end link. To position a regional structure in the task space, 3 DOF is the minimum requirement. A 3-DOF regional structure with a 3-DOF wrist will compose a 6-DOF robot whose end-effector can point to any direction and position in any location in its workspace. More DOFs in the regional structure with the same wrist will create a redundant manipulator. Therefore, we begin with 3 DOF for the regional structure of a modular robot in the task specification.

Note that the number of DOFs sent to the enumeration procedure is the minimum DOF required for a task. A larger number of DOFs may increase the dexterity at the end link, but control and planning issues become more complicated. The number of DOFs specified in the structure description does not necessarily satisfy the task specifications. The reachability of the task points constrains the minimum number of DOFs. When all of the candidate assembly configurations for a given number of DOFs fail to satisfy the task specifications, the number of DOFs is increased by one and the enumeration procedure is started again. This process is repeated until all task points are within the reach of the robot.

6.3.2. Topology Selection

The topology of a robot is directly related to the fundamental function of the robot. Robots with different topologies perform different classes of tasks. For instance, a serial manipulator typically has a large workspace, which makes it suitable for pick-and-place work, while serial/parallel hybrid robots have limited workspace, which make

them suitable for fine motion manipulation. The control and planning issues in those systems are also different. Motion planning in a manipulator generally is planning the trajectory of the end-effector. For a robot hand, the planning of the fingertip locations on the grasped object becomes an important issue [12,13].

During the process of finding task-optimal configurations, the topology of the robots and the corresponding class of tasks are described in the structure and task specifications respectively. The robot topology is provided prior to the enumeration procedure in generating candidate assembly configurations with identical topology.

Because serial-type manipulators are widely used in industrial application, they are chosen as a template topology for the task-optimal configuration problem. The task defined in Section 6.2.1 is a set of working points in the operation space, which is a class of tasks serial manipulators can execute.

6.3.3. Module Assembly Preference

Due to the possibility of multiple connections, some joint assembly patterns will create kinematic constraint, such as link interference and joint redundancy, in a modular robot assembly. Link interference means collisions among different links in a robot due to undesirable joint motions. The inverse kinematics solution may not be realized because of the link interference. Joint redundancy occurs when the axes of two revolute joints attached to the same link are collinear as shown in Fig. 6.3. When a redundant joint assembly pattern is included in a robot assembly, the two joints create only one type of rotary motion along the joint axes; hence, the effective degree of freedom of the robot is decreased by one. Indeed, generically a robot will lose DOF at singular positions where the Jacobian matrix loses rank. Joint redundancy will always cause loss of Jacobian rank. This will cause difficulties in controlling the robot motion and finding inverse kinematics of the end-effector. However, redundant joints extend the dimension of the robot arm, and thus enlarge the workspace. Besides, pairs of redundant joints are fault tolerant [77]: when one of the two joints fails, the other joint can still drive that degree

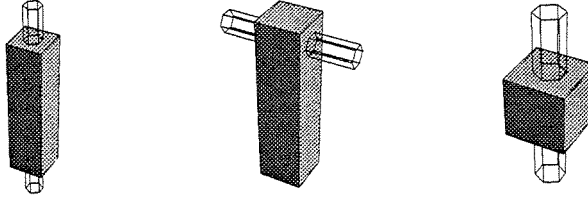


Figure 6.3: Assembly patterns with joint redundancy

of freedom. The decision to retain or to remove these conditions depends on the user's intention, which is described in the structure specifications.

In order to retain or filter out a joint assembly pattern in a modular robot, a binary-valued function called the *module assembly preference* (MAP), ϕ , will be defined on every distinct assembly pattern. Let \mathcal{F}/\mathcal{R} represent the set of distinct assembly patterns, where \mathcal{F} is the set of joint patterns and \mathcal{R} is the symmetry rotation group of that type of links.

Definition 6.2: A module assembly preference, $\phi : \mathcal{F}/\mathcal{R} \rightarrow \{0, 1\}$, is a surjective function such that

$$\phi : [f] \mapsto w, \quad w = 0, 1, \quad (6.2)$$

where $[f] \in \mathcal{F}/\mathcal{R}$ represents a distinct joint pattern.

$\phi([f]) = 0$ represents an undesirable joint pattern. The MAP will be stated in the structure specifications.

The preference of an entire robot assembly configuration can be defined based on the MAP for individual assembly patterns. This preference value will indicate the existence of undesirable joint patterns in the robot structure. Since a row vector in an AIM represents the assembly state on a link in the robot, the preference value on every link assembly can be obtained by applying a MAP to every row of the AIM. Suppose the robot has n links and the values of the MAP on Link i is w_i . The *structural preference* of an entire robot assembly configuration, Φ , is defined as follows.

Definition 6.3: The structural preference, Φ , of a modular robot assembly configuration A is the product of the individual MAPs of the robot's joint assembly patterns,

$$\Phi(A) = \prod_{i=1}^n w_i, \quad (6.3)$$

where A is the AIM of a robot assembly configuration.

Since w_i is equal to 0 or 1, $\Phi(A)$ is equal to 0 or 1 as well. $\Phi(A) = 0$ indicates the assembly configuration contains undesirable link assembly states which may cause link interference or link redundancy defined in the structure specifications. The choice of a MAP, ϕ , is illustrated in the following example.

Example 6.4: The structure specification of a modular robot assembly are given by:

- **DOF:** 3
- **Topology:** fixed base serial type with R-joints and prismatic links
- **Kinematic Constraints:** minimum link interference and no joint redundancy

The DOF and topology requirement will be sent to the enumeration procedure to generate all candidate configurations. In this serial manipulator, all links except the base and the end link are connected by two joints. From the joint assembly enumeration algorithm described in Section 4.1, we obtain nine distinct assembly patterns for an intermediate prism link. The choice of no joint redundancy implies that the MAP will be zero for all joint patterns in which two joint axes are collinear. The requirement on minimum link interference can be implemented if we set to zero the MAP of the patterns in which two joints are attached to the same end of the prism. The MAP of the rest of the patterns are all set to one.

For the end prism link, there are only two distinct assembly patterns. The requirements on the robot structure have little influence on the end link patterns because there is only one joint connected to the end link. Table 6.4 shows the MAP for assembly patterns on a prismatic link and a cubic link modules according to the kinematic constraints described above. Suppose

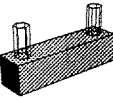
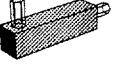
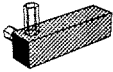


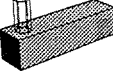
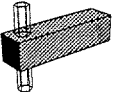

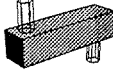
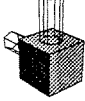
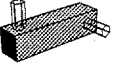

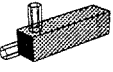
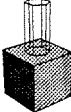
| ASSEMBLY PATTERNS | MAP | ASSEMBLY PATTERNS | MAP |
|---|-----|---|-----|
|  | 1 |  | 1 |
|  | 0 |  | 0 |
|  | 1 |  | 1 |
|  | 0 |  | 1 |
|  | 1 |  | 1 |
|  | 1 |  | 0 |
|  | 0 |  | 1 |

Figure 6.4: MAP for assembly patterns on a prism and a cube

$$A_1 = \begin{pmatrix} 1 & 0 & 0 & FB \\ 1 & 2 & 0 & L \\ 0 & 5 & 2 & L \\ 0 & 0 & 2 & L \\ R & R & R & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 0 & 0 & FB \\ 1 & 2 & 0 & L \\ 0 & 1 & 9 & L \\ 0 & 0 & 2 & L \\ R & R & R & 0 \end{pmatrix} \quad (6.4)$$

are the AIMs of a 3-DOF fixed base robot. The structural preference of A_1 is calculated starting from the base module. Since there is only one assembly state on the fixed base, its MAP is set to 1. There are two R-joints connected to port 1 and 2 of link 2, so $w_2 = 1$. R-joints are connected to port 2 and 5 of link 3, so $w_3 = 1$. For the end link,

$w_4 = 1$. Therefore, $\Phi(A_1) = w_2 \cdot w_3 \cdot w_4 = 1$. Similarly, in A_2 , $w_2 = 1$, $w_3 = 0$, and $w_4 = 1$, so $\Phi(A_2) = 0$. ■

The above example shows that kinematic constraints described in the robot structure specifications can be translated into a set of rules called module assembly preferences. The product of individual link MAPs forms the structural preference, which provides a convenient way to evaluate the robot assembly over the kinematic constraints.

6.4. Assembly Configuration Evaluation Function of Serial Robots

The structure of an ACEF for a serial modular robot is shown in Fig. 6.5. This function evaluates the “goodness” of a robot assembly configuration for a required task and structure specification. The “goodness” is represented by a non-negative real number. An AIM with large real value represents a good assembly configuration according to the requirements.

The input to the ACEF is an AIM with a predefined number of DOF and topology. The structure of an ACEF is divided into two parts: task and structure evaluations. The parameters related to robot structure is the MAP, ϕ , which is a direct interpretation of the kinematic constraints on the robot configuration. Given ϕ , the structural preference Φ of the AIM can be found according to (6.3). The part on task evaluation contains two sets of parameters: task points and TEC. Task points define the robot task and TEC is chosen from the available local performance measures mentioned in Section 6.2.2 depending on the objective of the task.

In the first part of task evaluation, the workspace check procedure checks a given AIM among all task points. This test determines the capability of an assembly configuration carrying out the specified task. The next section will explain the workspace check procedure in detail. If any one of the task points is outside of the robot’s workspace, there is no need to proceed with the evaluation of task performance, and the total task performance measure μ will be set to zero. If all task points are reachable, the TEC is

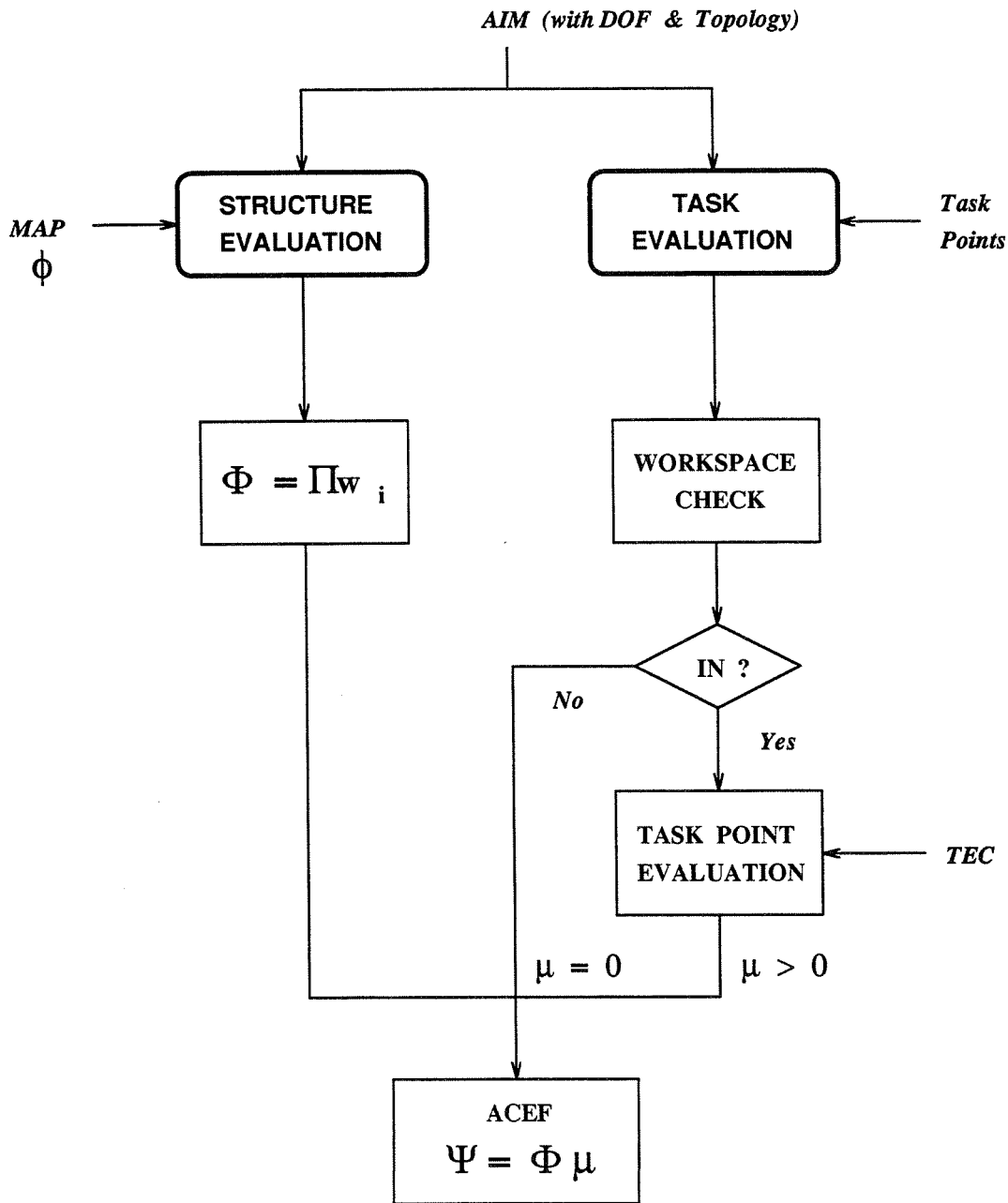


Figure 6.5: Structure of ACEF for serial modular robots

employed to calculate the total performance μ . The ACEF is defined to be the product of the structure preference Φ and the total performance μ .

Definition 6.5: The ACEF, Ψ , for the performance of a robot assembly, A , is

$$\Psi(A) = \Phi(A) \cdot \mu(A). \tag{6.5}$$

Since $\Phi(A)$ is equal to zero or one, and $\mu(A) \geq 0$, by definition, $\Psi(A) \geq 0$ as well. Note that the structure preference Φ functions as a filter for the ACEF. For assembly configurations satisfying the kinematic constraints, Φ is always equal to 1. Therefore, $\Psi(A) = \mu(A)$. Comparing the total performance of AIMS is equivalent to the comparison of the TECs of the task points only.

6.4.1. Workspace Check Procedure

An intuitive way to check whether a task point is in the workspace is to derive a set of close form algebraic expressions describing the boundary of workspace in terms of the robot parameters. Based on these expressions, a system of inequalities can be set up to define areas in the reach of the manipulator. However, the highly nonlinear geometry of a robot workspace [36,55,93,96,106] renders the formulation of this algebraic expression almost impossible.

A feasible way is to solve the inverse kinematics for the task point. The inverse kinematics of a general 6 DOF robot contains a set of highly nonlinear trigonometric equations which can be transformed into a set of multivariate polynomial equations. The solution to this set of polynomial equations contains real and complex roots. Real roots correspond to realizable robot postures for the task points, while complex roots are not realizable. If all roots from an inverse kinematics solution are complex, this task point is out of the manipulator's reach.

Here we adopt a similar concept, but use a numerical inverse kinematics technique during the workspace check. Using a numerical inverse to check the workspace can be very efficient and adaptable to robots with different DOFs. The solution of the numerical inverse kinematics is found if the calculated task point is within a preset tolerance ϵ in the neighborhood of the true task point. If the true task point is outside of the workspace, the calculated point will never converge to the true task point. The numerical inverse kinematics (NIK) mentioned in Section 5.3 is a very robust and efficient algorithm which will converge to a solution within a few iterations if the solution exists. Therefore, the workspace check is performed by setting a maximum

number of iterations for NIK initially. If the actual number of iteration exceeds the upper bound and no converging solution is found, this task point is considered out of reach by the robot.

Note that the speed of convergence of the NIK solution depends on the selection of the tolerance ϵ . When the task point is within the workspace but very close to its boundary, the solution of the NIK method will converge to the true point very slowly and the number of iterations may exceed the preset upper bound. However, the posture of the manipulator is usually very close to a singularity. Hence, the task point will be marked infeasible for this robot assembly.

6.5. Genetic Algorithms for Modular Robots

A “Genetic algorithm” (GA) is a search and/or optimization method based on the model of the ecological system in which the mechanics of natural selection and natural genetics are primary factors for improving the performance of a population of creatures. It was first developed by Holland and colleagues at the University of Michigan [40]. In this algorithm, candidate solutions are coded into string structures similar to the natural genetic code. A fitness function will assign a fitness value to every string. The surviving candidates, which are selected by the principle of survival of the fittest, are combined among themselves with a structured, yet randomized, information change to form a new generation of candidate solutions. The string structures of the new generation are created by using bits and pieces of the fittest in the previous generation. It is argued that these bits and pieces of the string, or *schemata*, are contributed to the overall performance of the string, and hence create offsprings with higher fitness values.

The most prominent feature of GAs is the use of a coded parameter set. A point in the search space is represented by a coded string. Furthermore, the algorithm starts from a set of points instead of one. A generic GA uses three operators to mimic the adaptive process of natural systems: (1) the reproduction operator, (2) the crossover operator,

and (3) the mutation operator. The reproduction operator is a process to select the survival in a set of candidate strings according to their fitness function values. The fitness function depends on the goal for the search/optimization problem. The fitness value determines the probability of a string contributing to the offspring in the next generation.

The crossover operator is a reform operation for the survival candidates. The survivors of the reproduction process will enter a mating pool to create a set of new strings. The mates are chosen randomly among strings in the pool. In natural systems, the mated creatures create a new generation by exchanging information between the two. In the same way, the crossover process is performed by exchanging bits or pieces of strings among the old surviving strings. The bits or pieces are crossed in couples by random selection.

In a natural system, mutation may drastically change the characteristics of a creature. In the artificial genetic approach, the mutation process result in escapes from the local minimum in search space. In the GA, the mutation operator randomly alternates the value of a string position during the crossover process with a very small probability.

Note that a GA uses an objective function (the fitness function) only; no auxiliary information, such as the gradient of the objective function, is required. And it uses probabilistic transition rules instead of deterministic ones. These features make the GA a robust search algorithm for a wide range of applications including both continuous and combinatorial optimization problems [35].

Several researchers have been applying GAs to modular or distributed robotic systems. A distributed GA is proposed by Ueyama, Fukuda, and Arai [95] for the structure configuration of CEBOT. The objective of this work is to plan paths between cells so that they can be connected together in a prescribed assembly sequence. Different paths that achieve the same goal are coded into string structure. The fitness function is the sum of the distance between any two cells. In the task-based robot design proposed by Kim and Khosla [48], a multiple-population GA (MGA) is employed to find an optimal

serial manipulator design that satisfies all design constraints.

The task-optimal configuration problem is a combinatorial optimization problem. This kind of problem also can be solved by using exhaustive search method. In the exhaustive search method, a search tree on AIMs is built. Applying standard Breath-First-Search or Depth-First-Search algorithms, a globally optimal solution can be found. For a small number of assembly configurations, this technique can find the solution in a reasonable amount of time. However, as the number of robot DOFs increases, the set of assembly configurations becomes factorially large and the exhaustive search becomes almost impossible. Furthermore, one can imagine that there is an analogy between the structure of gene and the structure of a serial-connected modular robot. The structure of a gene consists of a string of alleles. The characteristics of the entire gene is determined by arrangement of alleles, the building block. The performance of a modular robot is determined by the module assemblies that compose the entire kinematic chain. Every link-joint assembly contributes to the final characteristics of the robot. Therefore, GAs are a feasible approach for solving the task-optimal problem.

6.5.1. Coding Schemes for AIMs

In the task-optimal modular robot configuration problem, the parameter set is the set of AIMs. Therefore, a coding scheme is necessary to transform an AIM into a coded string for the application of GAs.

A binary string of 0 or 1 is chosen to represent an AIM because it can offer the maximum number of schemata per bit of any coding [35]. We call this binary string an *assembly string*. This string contains numerous substrings representing the types of every joint and link and joint patterns in an AIM. The lengths of the substrings are determined by the numbers in the type of joints and links and distinct joint patterns. Since joints and links are arranged in alternating sequence in a serial type robot, the substrings representing the types of joints and links are arranged in the same alternating sequence as in the AIM shown in Fig. 6.6.

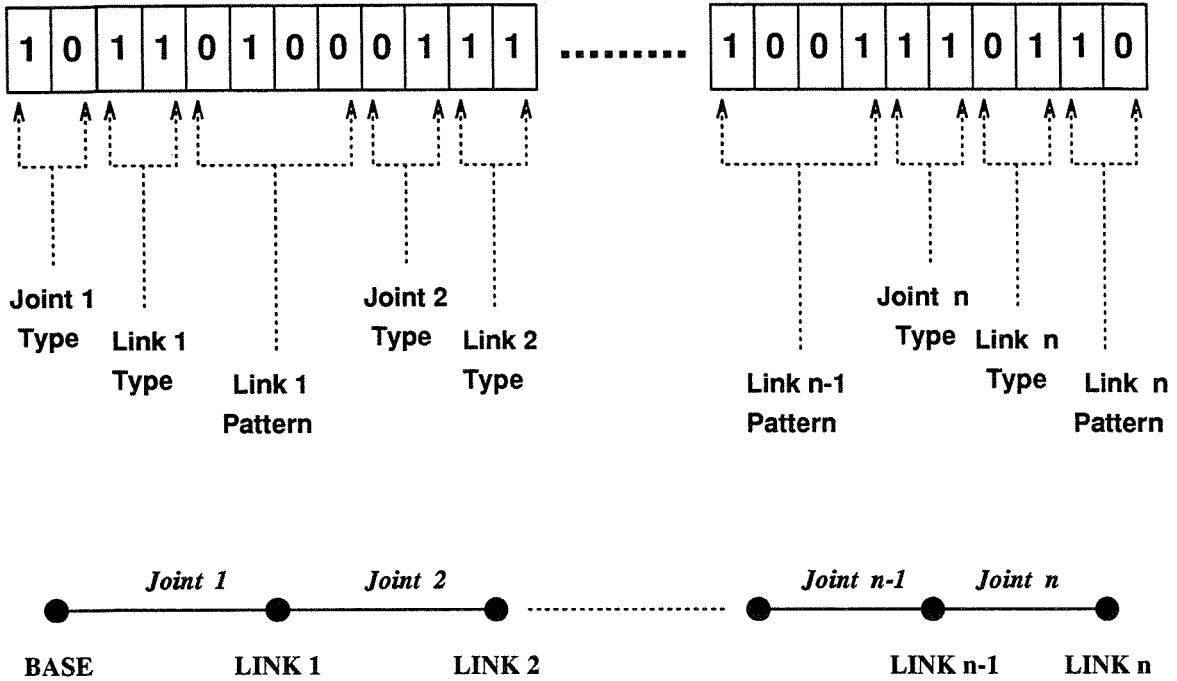


Figure 6.6: An assembly string representation of an AIM

Suppose the substring for the joint type is of length l_1 . This substring can represent 2^{l_1} types of joints. Each type of joints is assigned a unique bit string pattern of length l_1 . Similarly, a substring of the link type of length l_2 can represent up to 2^{l_2} types of links. In order to reduce the size of the search space, only a distinct joint pattern is assigned a bit string pattern; in other words, equivalent joint patterns have only one binary substring representation. If this joint pattern substring is of length l_3 , it can represent up to 2^{l_3} distinct joint patterns. Note that the number of joints connected to intermediate links in the serial robot, i.e., link 1 to $n - 1$ (Fig. 6.6), is different from that to the end link, i.e, link n . Therefore, the length of the joint pattern substring for the end link is different from that of the intermediate links, and we assume the length of this substring to be l_4 . Since there is only one method to attach a joint to the base of the robot, the joint pattern on the base is unnecessary to be specified in the assembly string. Hence, an n -DOF serial type robot can be expressed by an assembly string of length $n l_1 + n l_2 + (n - 1) l_3 + l_4$. If all joints (or all links) are the same, the joint (link) type substrings can be removed from the assembly string, i.e, $l_1 = 0$ (or $l_2 = 0$).

Because the number of distinct joint patterns is determined by the link module type, the joint pattern substring must be sufficiently long to represent the entire set of joint patterns. For some link module type, the number of distinct joint patterns which this substring can represent is much larger than that of the actual distinct joint patterns. Hence, it is always possible to map an AIM into an assembly string, but the converse does not hold every time. In other words, the mapping from the set of AIMS to the set of assembly strings is injective. For consistency, the fitness values of those assembly strings that cannot be mapped back to AIMS will be set to zero.

6.5.2. GA for Task-Optimal Configuration Problem

Fig. 6.7 depicts the application of GA in solving the task-optimal configuration problem. The input is a set of randomly chosen assembly strings, `PopInitial`. The number of elements in the set is specified by the user. The fitness function becomes the ACEF. The ACEF is a task dependent function, in which the task and structure specifications are the parameters. The fitness value of every AIM in `PopInitial` is obtained through an ACEF. If the fitness values of the initial set of strings are all equal to zero, the initial assembly string generation procedure will be repeated until a string with non-zero fitness value is found.

A new generation of AIMS are created by the three GA operators. In the reproduction process, the fitness values of all assembly strings in the population set are summed up. The portion of the fitness value of a string in the sum of fitness defines the probability of that string contributing to the next generation, i.e., the number of copies of the same string pattern appearing in the next generation. At the crossover stage, the pieces of the string for exchange are chosen randomly between each pair of assembly strings. The mutation operator is implemented as a process that alternates the values of the string position with a very small probability during the reproduction and crossover stages.

The new generation of assembly strings are presented to the ACEF for fitness evaluation, and the whole process will repeat until a predetermined generation, i.e., `PopFinal`

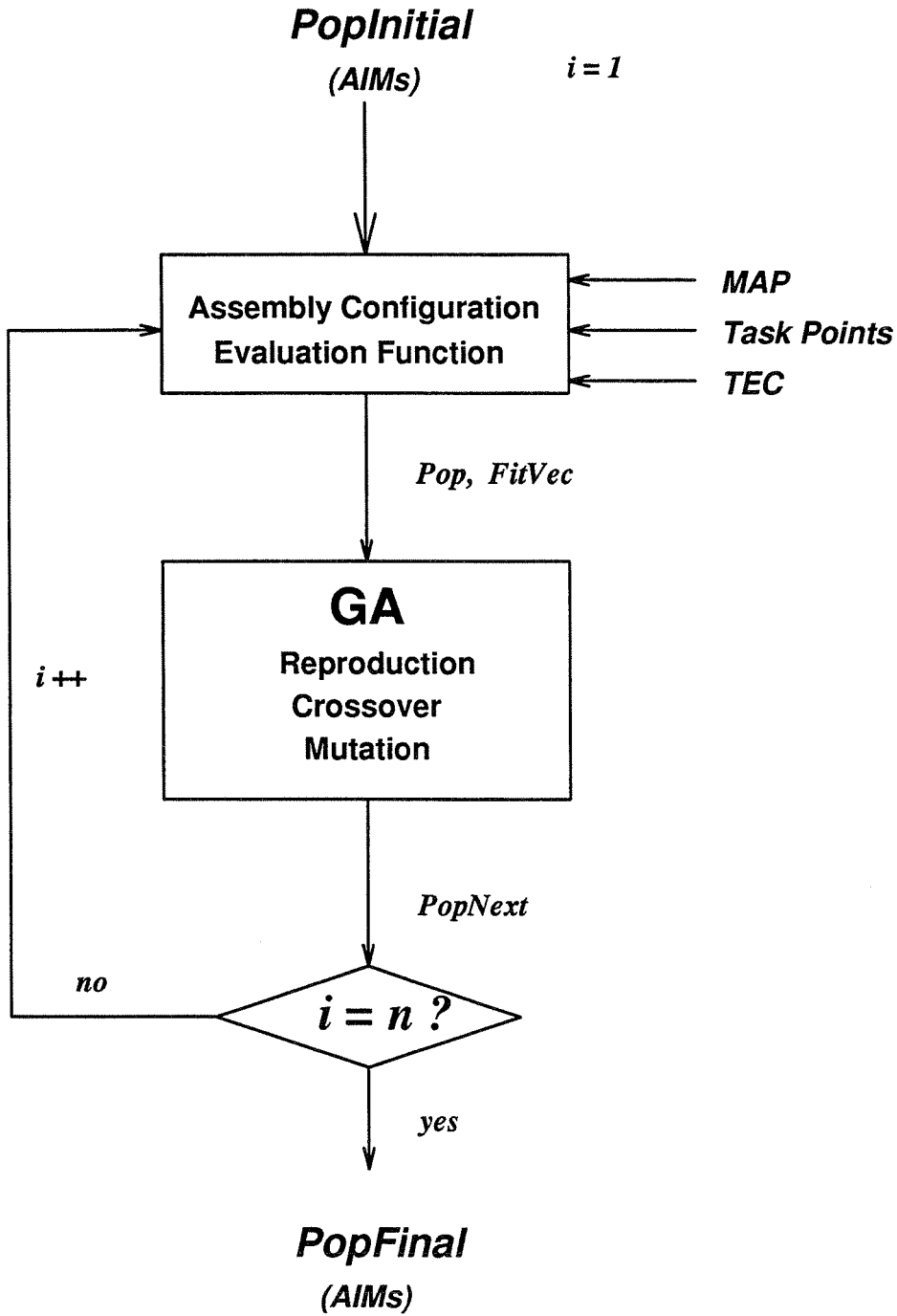


Figure 6.7: GA for task-optimal configurations

in the figure, is reached. After the destination generation is reached, we choose the assembly string in this generation that has the largest fitness value as the optimal assembly configuration satisfying the required task and structure specifications.

6.6. Examples

In this section, we demonstrate the use of genetic algorithms in solving a task-optimal configuration problem.

Example 6.6: We wish to find a 3-DOF fixed base serial robot with R-joints that passes through a set of task points listed in Table 6.2. We also require that there is no redundant joints and minimum link interference. The TEC is chosen to be the manipulator's manipulability measure. With those task and structure specifications on the robot configuration, the ACEF can be formulated using the MAP defined in Fig. 6.4 and the TEC of (6.1).

| Task Point | | Task Point | |
|------------|-----------------|------------|-----------------|
| 1 | (3.0, 0.5, 0.5) | 4 | (2.0, 2.0, 0.5) |
| 2 | (3.0, 0.5, 1.5) | 5 | (0.5, 3.0, 0.5) |
| 3 | (2.0, 2.0, 1.5) | 6 | (0.5, 3.0, 1.5) |

Table 6.2: Task point set (6.6)

The initial set of AIMS is shown in Fig. 6.8. The empty boxes represent assembly strings that do not correspond to any AIM. Their fitness values are set to zero. The fitness values of these AIMS obtained from the ACEF are (5.7963, 0., 0., 5.7963, 0., 0.). The parameters used in the GA are $P_{cross} = 0.6$, the probability of crossover operation, and $P_{mutate} = 0.1$, the probability of mutation. The destination generation is chosen to be the 10th generation. After evolving ten generations, the AIMS in the final generation are shown in Fig. 6.9. Their fitness values are (5.7963, 0., 5.7963, 5.7963, 0., 0.). Fig. 6.10 shows the average and maximum fitness value in every generation. Assembly configurations 1, 3, and 4 are identical and have the highest fitness value of 5.7963 so they are chosen as the optimal one. ■

Example 6.7: Now we construct a robot with a new set of task points listed in Table 6.3. The task points are on an 135° arc of $r = 5.656$. The task and structure specifications are identical to Example 6.6.

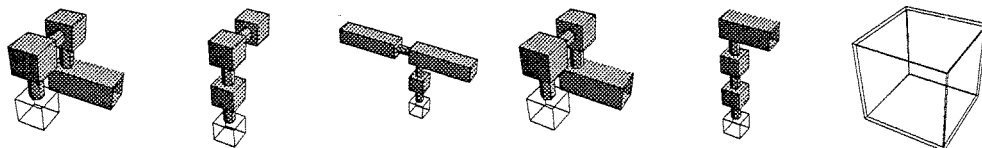


Figure 6.8: Initial configurations (6.6)

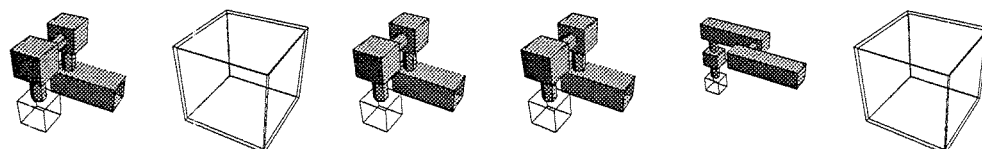


Figure 6.9: Final configurations (6.6)

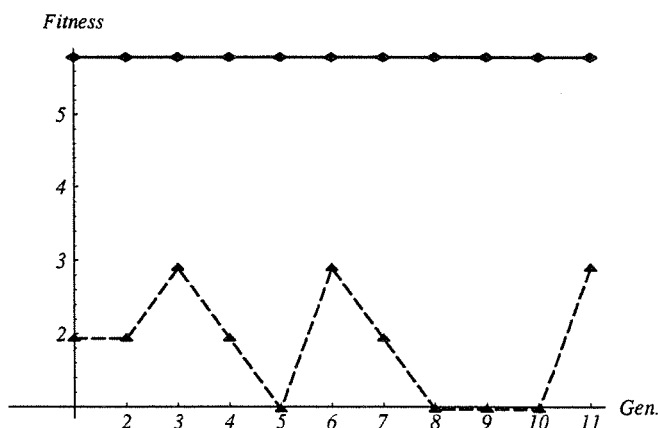


Figure 6.10: Average and maximum fitness in every generation (6.6)

| Task Point | | Task Point | |
|------------|----------------------|------------|---------------------|
| 1 | (-5.657, 0.0, 4.0) | 4 | (0.0, 5.657, 4.0) |
| 2 | (-4.899, 2.828, 4.0) | 5 | (2.828, 4.899, 4.0) |
| 3 | (-2.828, 4.899, 4.0) | 6 | (4.0, 4.0, 4.0) |

Table 6.3: Task point set (6.7)

The initial set of AIMS is shown in Fig. 6.11. Their fitness values obtained from the ACEF are (0., 0., 0., 0., 0., 25.45). The parameters used in the GA are $P_{cross} = 0.6$ and $P_{mutate} = 0.1$. The destination generation is chosen to be the 20th generation. After evolving 20 generations, the AIMS in the final generation are shown in Fig. 6.12. Their

fitness values are $(0., 0., 0., 25.45, 0., 0.)$. Fig. 6.13 shows the average and maximum fitness value in every generation. Assembly configuration 4 has the highest fitness value of 25.45 so it is chosen as the optimal one.

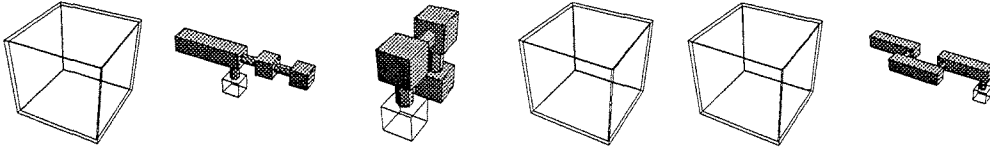


Figure 6.11: Initial configurations (6.7)

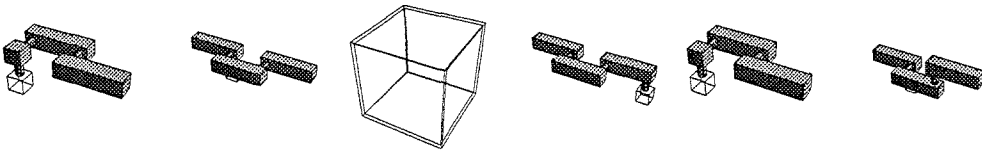


Figure 6.12: Final configurations (6.7)

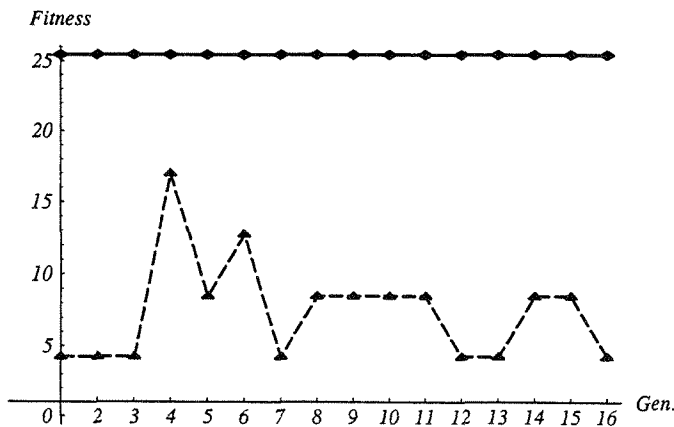


Figure 6.13: Average and maximum fitness in every generation (6.7)

Example 6.8: Now we use the same set of task points and the same task and structure specifications. The parameters used in the GA are $P_{cross} = 0.6$ and $P_{mutate} = 0.25$, which is different from the above example. The destination generation is chosen to be the 3th generation. The initial set of AIMs is shown in Fig. 6.14. Their fitness values

obtained from the ACEF are $(0., 0., 0., 25.45, 0., 0.)$. After evolving three generations, the AIMs in the final generation are shown in Fig. 6.15. Their fitness values are $(0., 0., 0., 25.45, 0., 0.)$. Fig. 6.16 shows the average and maximum fitness value in every generation. Assembly configuration 4 has the highest fitness value of 25.45 so it is chosen as the optimal one. This example shows that with a high mutation probability, an optimal assembly configuration can be obtained with fewer generation of evolution.

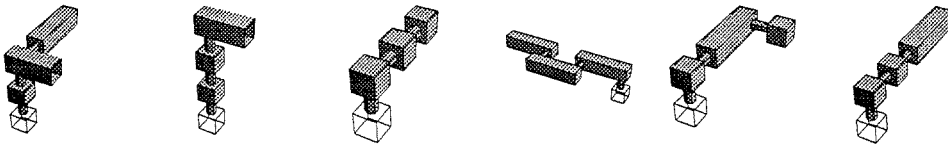


Figure 6.14: Initial configurations (6.8)

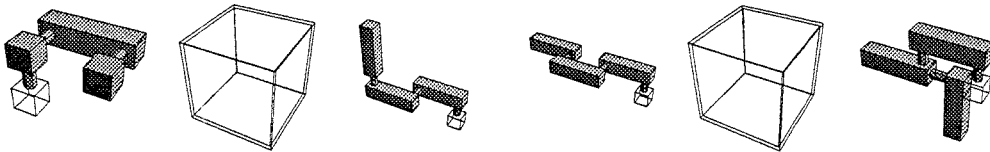


Figure 6.15: Final configurations (6.8)

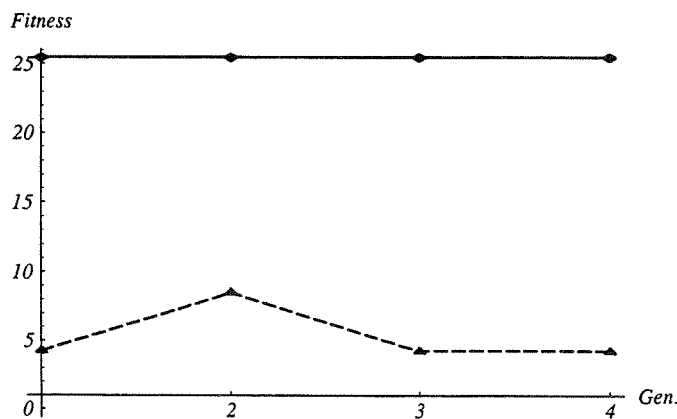


Figure 6.16: Average and maximum fitness in every generation (6.8)

6.7. Discussion

In this chapter, the task-oriented optimal configuration problem in modular robots is formulated as a combinatorial optimization problem. It is also shown that by categorizing the description of a robot task into task and structure specifications, one is able to construct an assembly configuration evaluation function that can evaluate the task performance of a robot while rejecting assembly patterns that lead to undesirable kinematics. Examples are demonstrated using genetic algorithms to solve this optimization problem. The genetic algorithm is a search algorithm that follows probabilistic rules based on the mechanics of natural selection and natural genetics. Using genetic algorithms, the optimal solution is not guaranteed, but suboptimal solutions can always be attained. This becomes a trade-off between efficiency and optimality.

Chapter 7

Planning Multifinger Hand Grasps

This chapter considers planning force-closure grasps on planar and spatial objects by a multifinger robot hand. A grasp is a set of finger contacts on the object surface. A force-closure grasp is a set of finger contacts such that any external force and moment exerted on the object can be balanced by the force and moment generated by fingers at the contact locations. When the object is grasped by a robot hand, it is necessary to maintain the stability of the object subject to external disturbance force and moment in order to prevent it from slippage. Furthermore, the stability of the object is required when dextrous finger manipulation involving “finger relocation” [58] and “finger gaits” [41] are considered. Due to finger joint limits and/or finger surface area limits, manipulation based on rolling or sliding of the finger tips can generate only relatively small changes in object orientation or displacement with respect to the base of fingers. To generate large displacements, it is often necessary to lift some fingers, relocate them on the object surface, and manipulate the object again using rolling or sliding contact, like the baton twirling example given by Fearing [29]. To reject disturbances in either relocating or gaiting phases, the fingers that remain in contact with the object must maintain force-closure. Further, in planning a complex finger gait, it is required that force closure is maintained during all gait transitions.

Only smooth curved shaped objects are considered here. The problem is divided into

two categories: two-finger grasps and n -finger grasps, where n is greater than 2. For two-finger grasps, a subset of force-closure grasps called *antipodal point grasps* is discussed. Antipodal points are a pair of points on an object whose normal vectors are collinear and in opposite direction. With appropriate types of finger contact (point contact with friction for planar objects or soft finger contact for spatial objects [61]), antipodal point grasps guarantee force-closure [65]. For more than two finger contacts, a general force-closure test on grasping planar objects is considered.

The analysis and planning of multifinger grasps has received considerable attention in the literature. Force-closure grasps on polygonal objects has been studied by many authors [60,74,81]. Nguyen [72] developed a geometric test for two-finger force-closure grasps on polygonal and polyhedral objects depending upon the relative locations of the two friction cones. Both Chen and Burdick [11] and Faverjon and Ponce [28] extend Nguyen's idea and developed a two-finger force-closure grasp test for planar curved shaped objects. Hong, et al [41]. first introduced the concept of antipodal point grasps on a smooth object. Their work was motivated by a heuristic approach to planning "finger gaits" in which fingers are placed on or in the neighborhood of antipodal points during the finger repositioning phases of a finger gait. Using a "distance function" on the distance between two contact points, they showed the existence of at least a pair of antipodal points on any smoothly shaped object.

This chapter introduces an extension of their result to *nonconvex* smooth objects. Some practical issues in implementing antipodal point finding algorithms, including optimization techniques and object modeling methods, are considered as well. A *grasping energy function* which is proportional to the square of distance between the two finger contacts is defined. It will be shown that critical points of this energy function which lie in the *force-closure region* in the *contact configuration space* correspond to pairs of antipodal points on the object surface. For the case of convex bodies considered in [41], the critical points always lie in the force closure regions. However, for the case of nonconvex bodies studied here, the critical points of this function may not lie in the force closure region. Thus, the search for antipodal points can be reduced to a

constrained optimization procedure.

Nevertheless, these methods cannot be generalized to three or more finger contacts. It is intuitively clear that finger gaiting operations will require more than the minimum number of contacts, or fingers, which are normally required for force-closure. For point contact with friction, two finger contacts are necessary for planar force closure. However, at least three fingers are required for gaiting around planar objects. A quantitative test for n -finger force closure grasp of a polygon, based on linear programming, has been recently proposed by [14].

A “qualitative” force-closure test for n ($n \geq 3$) finger contacts on a planar object is defined here. This test is based on the convex hull formed by the friction cone edge wrenches produced by every contact. This force closure test is termed “qualitative” because the test is binary. That is, it returns a true value if a grasp is force closure. Otherwise it returns a false value. It does not determine the optimality of a given grasp configuration with respect to given measure. Such a test would be called “quantitative.”

The structure of this chapter is as follows. Section 7.1 introduces the contact configuration space that represents all grasps on an object. Section 7.2 discusses the conditions for force-closure grasps on planar and spatial objects. Section 7.3 investigates two-finger antipodal point grasps on planar and spatial objects. By using a grasping energy function, finding antipodal point grasps on an object becomes a constraint global optimization problem. A newly developed global optimization scheme termed TRUST is employed. Examples are demonstrated for both planar and spatial objects. Section 7.4 formulates the force-closure test for n -finger grasps. Characteristics of force-closure regions in the contact configuration space are discussed. Applications of this force-closure test to multifinger manipulation and finger gait planning are demonstrated.

7.1. Contact Configuration Space

In robot motion planning, the position and orientation of a robot relative to a fixed reference frame is called a configuration [54]. The space of all configurations of the robot

is called a configuration space (C-space). Any finite and continuous robot motion will trace out a curve segment in the C-space. Thus, motion planning is equivalent to planning paths in the C-space of a robot. Applying this C-space concept to multifinger hand grasp planning, the locations of the finger contact on the grasped object can be called a *contact configuration*, or a *grasp*. The space of all grasps is called a *contact configuration space* (contact C-space). A grasp is represented by a point in the contact C-space. A continuous manipulation task performed by the fingers will trace out a trajectory in the contact C-space. Because a contact involves the interaction of two rigid bodies (a finger and a grasped object), the dimension of the contact C-space depends on both the number of finger contacts and the dimension of the object surface.

The case of 2-D objects is considered first. Assume that the boundary of a grasped object is a smooth and closed curve. Attach a coordinate frame, O , to the object. The object boundary can be described by a 1-to-1 parametric function:

$$\mathbf{p}(u) = [x(u), y(u)]^T, \quad u \in \mathbb{S}^1 \quad (7.1)$$

where u is called a *contact variable* and \mathbb{S}^1 is a unit circle. $\mathbf{p}(u_0) = [x(u_0), y(u_0)]^T$ represents a contact location, in frame O , on the object at u_0 . Since the object function is 1-to-1, u_0 can represent a contact, instead of $\mathbf{p}(u_0)$. We assume $\mathbf{p}(u)$ is at least once differentiable. Therefore, a unit tangent vector $\mathbf{t}(u_0)$, and a unit outward pointing normal vector $\mathbf{n}^+(u_0)$, exist at u_0 . The tangent vector is

$$\mathbf{t}(u_0) = \frac{\mathbf{p}'(u_0)}{\|\mathbf{p}'(u_0)\|} = [\hat{x}(u_0), \hat{y}(u_0)]^T, \quad (7.2)$$

where $\mathbf{p}'(u_0) = \frac{d\mathbf{p}}{du}|_{u_0}$. The outward normal vector $\mathbf{n}^+(u_0) = [-\hat{y}(u_0), \hat{x}(u_0)]^T$ if the parameter u is defined counter clockwise. The inward normal vector $\mathbf{n}^-(u_0) = -\mathbf{n}^+(u_0)$.

Definition 7.1: *Contact Configuration*

The n -tuple $\mathbf{q} = (u_1, \dots, u_n)$, which represents the location of n finger contacts on a planar object, with $u_i \neq u_j$, $i \neq j$, for $u_i \in \mathbb{S}^1, i = 1, \dots, n$, is called a *contact configuration* of an n -finger (or n -contact) grasp on a planar object. We call \mathbf{q} an *n -finger grasp* for abbreviation.

Note that contacts between the object and the robot hand may occur at other locations besides the fingertips, such as on the limb of a finger or on the palm of the hand.

Definition 7.2: *Contact Configuration Space for Planar Objects*

Let $\mathbb{T}^n = \overbrace{\mathbb{S}^1 \times \cdots \times \mathbb{S}^1}^{n \text{ times}}$ and $\Delta_{ij} = \{(u_1, \dots, u_n) | u_i = u_j, i \neq j, u_i \in \mathbb{S}^1\}$. The set

$$\mathcal{C}_n = \mathbb{T}^n \setminus \left(\bigcup_{\substack{i,j=1 \\ i \neq j, i < j}}^n \Delta_{ij} \right) \quad (7.3)$$

is called the n-contact configuration space (or n-contact C-space). Δ_{ij} represents all physically unrealizable contact configurations in which two fingers occupy the same location on the object. \mathcal{C}_n represents all possible n-finger grasps on the object.

For a 3-D object that is devoid of holes and homeomorphic to a sphere, the object surface can be described by a 1-to-1 function:

$$\mathbf{s}(\mathbf{u}) = [x(\mathbf{u}), y(\mathbf{u}), z(\mathbf{u})]^T, \quad (7.4)$$

where $\mathbf{u} \in \mathbb{S}^2$ is a contact variable. Since the function is also 1-to-1, we can use \mathbf{u}_0 instead of $\mathbf{s}(\mathbf{u}_0)$ to represent a contact point on the object. Analogous to Definition 7.1, a contact configuration of an n-finger grasp on a 3-D object is defined to be $\mathbf{q} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n)$. Let $\Gamma_{ij} = \{(\mathbf{u}_1, \dots, \mathbf{u}_n) | \mathbf{u}_i = \mathbf{u}_j, i \neq j, \mathbf{u}_i \in \mathbb{S}^2\}$ represents the set of unrealizable contacts on a 3-D object. The contact configuration space for 3-D objects becomes

$$\mathcal{D}_n = \overbrace{\mathbb{S}^2 \times \cdots \times \mathbb{S}^2}^{n \text{ times}} \setminus \left(\bigcup_{\substack{i,j=1 \\ i \neq j, i < j}}^n \Gamma_{ij} \right). \quad (7.5)$$

7.2. Force-Closure Grasps

A force-closure (FC) grasp is a set of finger contact locations, i.e, a contact configuration, on an object such that any external force and moment exerted on the grasped object can be balanced by the force and moment exerted by fingers at the contact locations. In a sense, it is a stable grasp since it can reject disturbance forces applied to

the object during a grasping or manipulation task. Because the local geometry of the contact point influences the possible directions of the applied finger contact force, the contact configuration of a force-closure grasp is closely related to the object geometry.

Typically, a finger cannot exert force in any direction on the contact surface. The types of contact determine the range of the finger force direction and the minimum number of fingers required for a force-closure grasp [61]. The simplest type of contact is *frictionless point contact* in which no friction exists between the fingertip and the object. The finger can apply forces normal to the object surface only.

In order to incorporate friction in a finger contact, *Coulomb* friction model is used. Coulomb friction model asserts that the allowed tangential force is proportional to the applied normal force. The constant of proportionality, which is termed the *friction coefficient*, is a function of the material that are in contact. Let f_n and f_t denote the magnitude of the applied normal force and tangential force respectively. Coulomb's law says that slippage occurs when $|f_t| > \mu f_n$, where μ is the friction coefficient. This model defines a conical region, termed a *friction cone*, at a contact location such that the allowable finger contact force must lie within the cone to prevent slipping. The range of the friction cone is determined by the friction coefficient μ .

A *point contact with friction* model is employed when friction exists between fingertips and the object. The fingertip can apply force within the friction cone without slipping.

A *soft finger* contact is a more realistic one in which not only friction forces but also torques about the normal are allowed on the contact surface. For simplicity, the torque is limited by a torsional friction coefficient.

7.2.1. Two-Finger Force-Closure Grasps on Planar Objects

For 2-D object grasping, we assume: (1) a point contact with friction model, and (2) a constant friction coefficient, μ , everywhere on the object. The friction cone at a contact point becomes a sector for planar objects. Sliding between the finger and the object will not occur as long as the finger force lies in the friction sector.

Denote the friction sector at a finger contact $\mathbf{p}(u_i)$ by $\mathcal{S}(u_i)$. The friction sector consists of two parts: one pointing outside of the object termed the *positive friction sector*, $\mathcal{S}^+(u_i)$, and the other pointing inward termed the *negative friction sector*, $\mathcal{S}^-(u_i)$, as shown in Fig. 7.1.

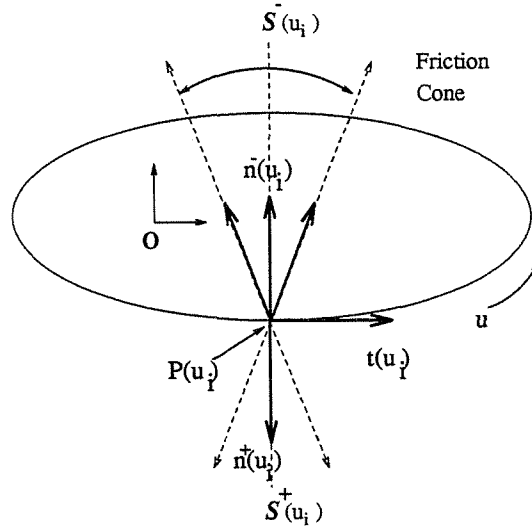


Figure 7.1: Friction cone at contact u_i

A force exerted by the finger contact at u_i is called *feasible* if it lies in the negative friction cone $\mathcal{S}^-(u_i)$. A two-finger grasp is said to be force-closure if any external force and moment can be balanced by a positive linear combination of two feasible contact forces exerted by the fingers [61]. A simple geometric statement proposed by Nguyen [72] says that a two-finger force-closure grasp, $\mathbf{q} = (u_1, u_2)$, can be achieved if and only if the line connecting contact points, $\mathbf{p}(u_1)$ and $\mathbf{p}(u_2)$, lies inside both $\mathcal{S}(u_1)$ and $\mathcal{S}(u_2)$. Since a friction sector contains a positive and a negative sector, there are two possible conditions satisfying this statement:

Definition 7.3: *Squeezing and Expanding Force-Closure Grasps*

Denote the line connecting contact points u_1 and u_2 by $\overline{u_1 u_2}$. If $\overline{u_1 u_2}$ falls inside both $\mathcal{S}^-(u_1)$ and $\mathcal{S}^-(u_2)$, the grasp \mathbf{q} is called a *squeezing grasp* (Fig. 7.2). If $\overline{u_1 u_2}$ falls inside both $\mathcal{S}^+(u_1)$ and $\mathcal{S}^+(u_2)$, the grasp \mathbf{q} is called an *expanding grasp* (Fig. 7.3).

Convex objects can be grasped by squeezing grasps only. Non-convex objects can be grasped by squeezing and possibly expanding grasps as well.

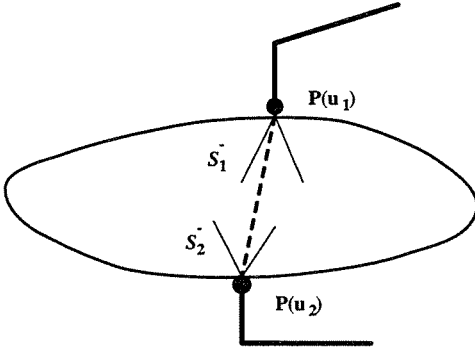


Figure 7.2: Squeezing grasp

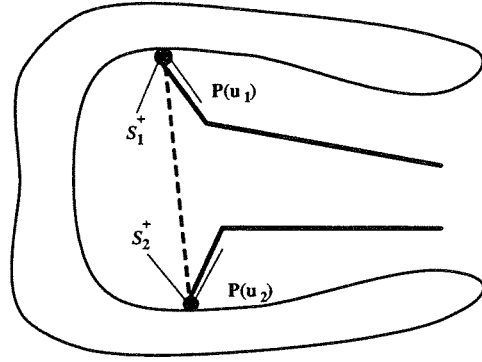


Figure 7.3: Expanding grasp

A squeezing force-closure grasp satisfies:

$$\mathbf{n}^+(u_1) \cdot \frac{\mathbf{p}(u_1) - \mathbf{p}(u_2)}{\|\mathbf{p}(u_1) - \mathbf{p}(u_2)\|} > c_f, \quad (7.6)$$

$$\mathbf{n}^+(u_2) \cdot \frac{\mathbf{p}(u_2) - \mathbf{p}(u_1)}{\|\mathbf{p}(u_2) - \mathbf{p}(u_1)\|} > c_f, \quad (7.7)$$

while an expanding grasp satisfies:

$$\mathbf{n}^+(u_1) \cdot \frac{\mathbf{p}(u_1) - \mathbf{p}(u_2)}{\|\mathbf{p}(u_1) - \mathbf{p}(u_2)\|} < -c_f, \quad (7.8)$$

$$\mathbf{n}^+(u_2) \cdot \frac{\mathbf{p}(u_2) - \mathbf{p}(u_1)}{\|\mathbf{p}(u_2) - \mathbf{p}(u_1)\|} < -c_f, \quad (7.9)$$

where $c_f = \cos(\tan^{-1}\mu)$.

7.2.2. N-Finger Force-Closure Grasps on Planar Objects

The geometric statement of a two-finger force-closure grasp mentioned in previous section cannot be generalized to three or more finger contacts. A general approach based on the convex hull of the friction sector edge wrenches produced at n finger contacts is adopted here.

The contact force exerted by the fingers on a planar object, along with its associated moment about the origin of O , is called a *contact wrench*, and can be represented as

a 3×1 vector \mathbf{w} . The space of all contact wrenches is termed the *wrench space*. It is isomorphic to \mathbb{R}^3 .

Let \mathbf{f}_i be the contact force exerted by the i^{th} finger at contact point u_i within the negative friction sector. Let \mathbf{f}_i^+ and \mathbf{f}_i^- be edge vectors of $\mathcal{S}^-(u_i)$ as shown in Fig. 7.1. The edge vectors can be written as a linear combination of the unit tangent and unit inward normal vector: $\mathbf{f}_i^\pm = \mathbf{n}^-(u_i) \pm \mu \mathbf{t}(u_i)$, where μ is the friction coefficient and is always greater than zero. Then \mathbf{f}_i can be expressed as a positive linear combination of the edge force vectors: $\mathbf{f}_i = \alpha_i^+ \mathbf{f}_i^+ + \alpha_i^- \mathbf{f}_i^-$, where $\alpha_i^\pm > 0$. Let the wrenches generated by the edge vectors of the friction cone be termed *edge wrenches*. By linearity, the contact wrench \mathbf{w}_i generated by \mathbf{f}_i can be expressed as a positive linear combination of the edge wrenches. It is also called a *feasible* contact wrench.

Let \mathbf{f}_e, τ_e be an external force and moment (also called an *external wrench*, $\mathbf{w}_e = [\mathbf{f}_e^T \ \tau_e]^T$) exerted on the object with respect to O . The force-closure condition requires that any external wrench can be counter-balanced by a positive linear combination of a set of feasible contact wrenches. Equivalently, it can be expressed as a positive linear combination of the edge wrenches:

$$\mathbf{f}_e = \sum_{i=1}^n \mathbf{f}_i = \sum_{i=1}^n \alpha_i^+ \mathbf{f}_i^+ + \alpha_i^- \mathbf{f}_i^- \quad (7.10)$$

$$\tau_e = \sum_{i=1}^n \mathbf{p}_i \otimes \mathbf{f}_i = \sum_{i=1}^n \alpha_i^+ \mathbf{p}_i \otimes \mathbf{f}_i^+ + \alpha_i^- \mathbf{p}_i \otimes \mathbf{f}_i^- \quad (7.11)$$

where $\mathbf{a} \otimes \mathbf{b} = a_1 b_2 - a_2 b_1$, for $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$. Combining (7.10) and (7.11), we obtain

$$\mathbf{w}_e = W\mathbf{c}, \quad (7.12)$$

where

$$\begin{aligned} \mathbf{w}_e &= [\mathbf{f}_e^T, \tau_e]^T \\ W &= \begin{bmatrix} \mathbf{f}_1^+ & \mathbf{f}_1^- & \dots & \mathbf{f}_n^+ & \mathbf{f}_n^- \\ \mathbf{p}_1 \otimes \mathbf{f}_1^+ & \mathbf{p}_1 \otimes \mathbf{f}_1^- & \dots & \mathbf{p}_n \otimes \mathbf{f}_n^+ & \mathbf{p}_n \otimes \mathbf{f}_n^- \end{bmatrix} \\ &= [\mathbf{w}_1(u_1) \ \mathbf{w}_2(u_1) \ \mathbf{w}_3(u_2) \ \mathbf{w}_4(u_2) \ \dots \ \mathbf{w}_{2n-1}(u_n) \ \mathbf{w}_{2n}(u_n)] \\ \mathbf{c} &= [\alpha_1^+, \alpha_1^-, \dots, \alpha_n^+, \alpha_n^-]^T. \end{aligned} \quad (7.13)$$

The $3 \times 2n$ matrix, W , is termed a *grasp map* in [69]. Its column vectors are edge wrenches of the contacts, and \mathbf{c} is a $2n$ -vector representing the magnitudes of those edge wrenches. Note that W is a function of the contact configuration: $W = W(\mathbf{q})$. An n -finger force-closure grasp requires that the $2n$ column vectors of $W(\mathbf{q})$ positively span the wrench space \mathbb{R}^3 .

Convex Hull

Let $X = \{x_1, \dots, x_m\} \subset \mathbb{R}^n$ be a finite set; the convex hull of X , denoted by $CO(X)$, is a set of all convex combination of all elements in X , i.e.,

$$CO(X) = \{a_1x_1 + \dots + a_mx_m \mid a_i \geq 0, \sum_{i=1}^m a_i = 1, x_i \in X\}$$

A hyperplane in \mathbb{R}^n divides \mathbb{R}^n into two half spaces. A hyperplane is said to be a *supporting hyperplane* of $CO(X)$ if it contains a boundary point of $CO(X)$ and $CO(X)$ is contained in one of the two closed half spaces determined by it.

Conditions on N-Finger FC Grasps

Denoting the column vectors of the grasp map W by \mathbf{w}_i , $i = 1, \dots, 2n$, and, $CO(W)$, the convex hull in wrench space \mathbb{R}^3 formed by the edge wrenches $\{\mathbf{w}_i\}$, we have the following proposition [69]:

Proposition 7.4: For planar object grasping that assumes a point contact with friction model, the following are equivalent:

1. An n -finger grasp $\mathbf{q} = (u_1, \dots, u_n)$ is force-closure.
2. \mathbf{w}_i , $i = 1, \dots, 2n$, of $W(\mathbf{q})$ positively span the wrench space \mathbb{R}^3 .
3. Let E_{ij} be the plane passing through the origin of the wrench space and containing \mathbf{w}_i and \mathbf{w}_j , where $i \neq j$ and $i, j = 1, \dots, 2n$. None of the planes E_{ij} is a supporting plane of $CO(W(\mathbf{q}))$.
4. $CO(W(\mathbf{q}))$ contains a neighborhood of the origin of \mathbb{R}^3 .

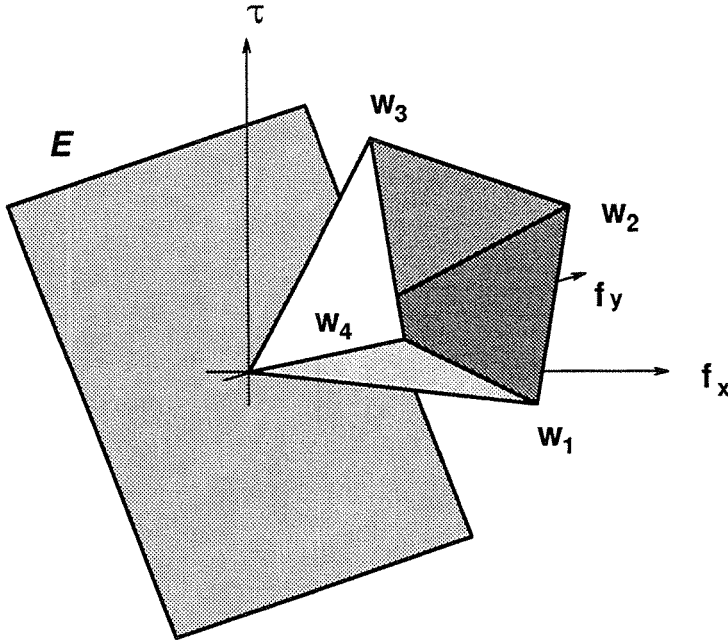


Figure 7.4: Supporting plane of $CO(W(\mathbf{q}))$

Fig. 7.4 shows the relationship of the convex hull, $CO(W(\mathbf{q}))$, and a plane E passing through the origin. If E becomes a supporting plane of $CO(W(\mathbf{q}))$, edge wrenches \mathbf{w}_i all lay in the same side of plane E . \mathbf{w}_i cannot positively span the wrench space.

7.2.3. Force-Closure Grasps on Spatial Objects

For 3-D object grasping, we assume: (1) a soft-finger contact model; (2) a constant friction coefficient μ ; (3) a constant torsional friction coefficient γ . We call the friction cone extending outside of the object $\mathcal{S}^+(\mathbf{u}_0)$ and the one inside the object $\mathcal{S}^-(\mathbf{u}_0)$. Nguyen [72] also showed that for 3-D object, a two-finger grasp, $\mathbf{q} = (\mathbf{u}_1, \mathbf{u}_2)$, with soft-finger contact model will be force-closure if and only if the line connecting the $\mathbf{p}(\mathbf{u}_1)$ and $\mathbf{p}(\mathbf{u}_2)$ lies strictly inside both friction cones $\mathcal{S}^-(\mathbf{u}_1)$ and $\mathcal{S}^-(\mathbf{u}_2)$ (or $\mathcal{S}^+(\mathbf{u}_1)$ and $\mathcal{S}^+(\mathbf{u}_2)$).

7.3. Two-Finger Grasp Planning

This section considers two-finger antipodal point grasps on planar and spatial smooth objects. Antipodal points are a pair of points on an object whose normal vectors

are collinear and in opposite direction. With appropriate finger contact conditions, antipodal point grasps guarantee force-closure [65,72].

Definition 7.5: *Antipodal Points for Planar Objects [53]*

Two contact points, u_1 and u_2 , on an object satisfying the following conditions are called *antipodal points*:

$$[\mathbf{p}(u_1) - \mathbf{p}(u_2)] \cdot \mathbf{t}(u_1) = 0 \quad (7.14)$$

$$[\mathbf{p}(u_2) - \mathbf{p}(u_1)] \cdot \mathbf{t}(u_2) = 0 \quad (7.15)$$

$$\mathbf{n}^+(u_1) + \mathbf{n}^+(u_2) = 0. \quad (7.16)$$

The definition of antipodal points on a 3-D object is similar to (7.14) to (7.16), except the tangent vectors $\mathbf{t}(u_1)$ and $\mathbf{t}(u_2)$ can be any vectors on the tangent planes at u_1 and u_2 .

7.3.1. Planar Objects

7.3.1.1. Force-Closure Regions in Contact C-Space

The inequalities (7.6), (7.7), (7.8), and (7.9) define regions in the contact C-space, \mathcal{C}_2 , called *force-closure regions* (FC-regions), or feasible grasping regions [11], in which the grasps are force-closure. Let \mathcal{F}^- denote all grasps in \mathcal{C}_2 satisfying (7.6) and (7.7) and \mathcal{F}^+ the grasps satisfying (7.8) and (7.9). $\mathcal{F} = \mathcal{F}^- \cup \mathcal{F}^+$ (where $\mathcal{F}^- \cap \mathcal{F}^+ = \emptyset$) is the set of all force-closure grasps. The force-closure curves (FC-curves) that bound the FC-regions in \mathcal{C}_2 are the zero sets of the functions obtained by replacing the inequalities (7.6) to (7.9) by equalities. Note that this method of finding FC-regions extends to 3-D case, whereas the method of Faverjon and Ponce [28], which relies upon the cross product of the friction cone normal and edge vectors, cannot. A 3-D object friction cone cannot be described by a linear combination of finite vectors; hence, the cross product method is no longer valid.

7.3.1.2. A Grasping Energy Function

A “grasping energy function” $E: \mathcal{C}_2 \rightarrow \mathbb{R}$ is defined as:

$$E(u_1, u_2) = \frac{1}{2} \kappa \|\mathbf{p}(u_1) - \mathbf{p}(u_2)\|^2 \quad (7.17)$$

E can be interpreted as the energy of a spring, with spring constant κ , connecting u_1 and u_2 . E is continuous and once differentiable.

Proposition 7.6: Antipodal points, u_1 and u_2 , correspond to a critical point of E . Conversely, only critical points of E lying in \mathcal{F} correspond to antipodal points.

Proof: Let (u_1, u_2) be antipodal points and substitute (7.2) into (7.14) and (7.15).

$$[\mathbf{p}(u_1) - \mathbf{p}(u_2)] \cdot \frac{\mathbf{p}'(u_1)}{\|\mathbf{p}'(u_1)\|} = 0 \quad (7.18)$$

$$[\mathbf{p}(u_2) - \mathbf{p}(u_1)] \cdot \frac{\mathbf{p}'(u_2)}{\|\mathbf{p}'(u_2)\|} = 0 \quad (7.19)$$

Since $\|\mathbf{p}'(u)\| \neq 0$, by smoothness of the boundary curve,

$$\kappa [\mathbf{p}(u_1) - \mathbf{p}(u_2)] \cdot \mathbf{p}'(u_1) = \frac{\partial E}{\partial u_1} = 0 \quad (7.20)$$

$$\kappa [\mathbf{p}(u_2) - \mathbf{p}(u_1)] \cdot \mathbf{p}'(u_2) = \frac{\partial E}{\partial u_2} = 0. \quad (7.21)$$

Thus, $\mathbf{q} = (u_1, u_2)$ is a critical point of E . Conversely, for nonconvex objects, a critical point of E does not necessarily satisfy (7.16) and is therefore not necessarily an antipodal point. Any grasp, \mathbf{q} , lying in \mathcal{F} , satisfies either (7.6) and (7.7) or (7.8) and (7.9). Note that for any $(u_1, u_2) \in \mathcal{F}$, the angle between two outward normal vectors $\mathbf{n}^+(u_1)$ and $\mathbf{n}^+(u_2)$ is between $(\pi - 2 \tan^{-1} \mu)$ and π . If $\mathbf{q}^* = (u_1^*, u_2^*)$ is a critical point of E and $\mathbf{q}^* \in \mathcal{F}$, then $\mathbf{n}^+(u_1^*)$, $\mathbf{n}^+(u_2^*)$ will be collinear and in opposite directions. Hence, (7.14), (7.15), and (7.16) are all satisfied. \mathbf{q}^* represents an antipodal point grasp on the object. ■

Since E is differentiable over \mathcal{C}_2 (which is compact), E must achieve both a maximum, at \mathbf{q}_{max} , and a minimum, at \mathbf{q}_{min} , in \mathcal{C}_2 . \mathbf{q}_{max} and \mathbf{q}_{min} are respectively termed the “maximal” and “minimal” grasps. Nonconvex objects can have critical points which are local minima and maxima of E . The properties of these critical points depend

upon the local object geometry near the antipodal points. Let $\mathbf{q}^* = (u_1^*, u_2^*)$ denote an antipodal point pair. If the object is convex (concave) at u_1^* and u_2^* , $E(u_1^*, u_2^*)$ will be a local maximum (minimum). If the object is convex at one antipodal point and concave at the other, the critical point may be a saddle point, local minimum, or local maxima, depending on the relative object curvature at u_1^* and u_2^* . E is necessarily convex (resp. concave) at the maximal (minimal) antipodal points.

These local properties can be used to differentiate between different antipodal grasp choices. The previous and ensuing discussion neglects the three dimensional volumetric properties of the fingertips. An automated grasp planner should check for interference between the finger and the object. If the grasping fingertips are convex, the maximal grasp does not require additional calculations which check for geometric interference. In principle, simple parallel jaw grippers could be used to grasp at the maximal grasp without complex calculations. However, it is known that the maximal two-finger grasp may be less “stable” [67] than other antipodal point grasps. Conversely, the minimal grasp is a more stable or immobile grasp. It may be more desirable, even if additional computations are required to check for interference between the object and fingers. The choice between locally maximal or minimal antipodal point grasps is thus a function of other task requirements.

7.3.1.3. Planning Antipodal Point Grasps

Proposition 7.6 suggests that antipodal points can be found by searching for the critical points of E in the FC regions. For many practical applications, we are more interested in finding the subset of critical points which are either minima or maxima of E . In these cases, we can the formulate antipodal point search problem as a constrained optimization problem:

$$\begin{aligned}
 &\text{maximize} && E^*(u_1, u_2) = \frac{1}{2} \kappa \|\mathbf{p}(u_1) - \mathbf{p}(u_2)\|^2 \\
 &\text{subject to} && (u_1, u_2) \in \mathcal{F} \subset \mathcal{C}_2
 \end{aligned}
 \tag{7.22}$$

where $E(u_1, u_2) = E^*(u_1, u_2)$, if one is interested in locally maximal grasps. The grasping energy $E(u_1, u_2) = -E^*(u_1, u_2)$ if locally minimal grasps are of interest. Any suitable constrained optimization method, such as the method of Lagrange multipliers, constrained Newton or quasi-Newton methods, or successive quadratic programming [6], can be used to solve (7.22). In most cases, the constraints arise only from force closure. However, in some practical cases, portions of the object surface may be occluded by nearby objects, or not visible to a robot vision system which generates object models. In such cases, additional constraints can be added to exclude these regions in the optimization process.

The aforementioned optimization methods have only local convergence properties. Thus, the antipodal points found by these methods will depend on the procedure's initial conditions. Multiple random start methods [8] can be used to find all of the local critical points. Alternatively, constrained global optimization techniques, such as simulated annealing [49] or interval analysis methods [68] can be used to find the globally minimal or maximal grasp. In this work, the global maximum of E is found using a recently developed global optimization algorithm, termed TRUST (see [10] for details). This method is simple to implement and has been found to be substantially faster than other global optimization methods in benchmark tests. TRUST uses a novel "tunneling" method which finds the global extrema by repeatedly escaping local extrema. Thus, on the way to finding the global solution, many local critical points, which correspond to feasible antipodal grasping points, are identified. However, all critical points are not found on the way to the global optimum.

These methods suggested above are most useful if the antipodal points are isolated in \mathcal{C}_2 . However, when the object contains parallel edges or faces, for example, the antipodal points will not be isolated point sets. In these cases, the critical points of E can be found using continuation techniques developed for numerical bifurcation analysis [45].

From a set of antipodal points, which are found using a multiple random start or as

intermediate steps of a tunneling global optimization, one can select an antipodal point pair based on additional considerations. For example, interference between fingertips, or a distance between antipodal points (which might exceed the greatest dimension of the hand workspace) can be used to cull antipodal points from the feasible set.

7.3.1.4. Representations of Planar Objects

Here we parameterize the boundary of a curved planar object by cubic B-spline curves that are frequently used in computer graphics applications. This method is computationally efficient, produces surface with satisfactory smoothness, and can be used to approximate nearly any smooth object with arbitrary precision. For more detailed treatment of B-spline curves, please refer to [104].

A cubic B-spline curve is a collection of piecewise continuous parametric cubic polynomial curve segments whose derivatives are continuous at “knot points.” If the parametric intervals in all segments are equal, the curve is called a *uniform cubic B-spline curve*. Every segment i in the uniform cubic B-spline curve has the following form:

$$\mathbf{p}_i(t_i) = \mathbf{a}_i t_i^3 + \mathbf{b}_i t_i^2 + \mathbf{c}_i t_i + \mathbf{d}_i, \quad (7.23)$$

where $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}_i \in \mathbb{R}^2$ are coefficients of the polynomial \mathbf{p}_i and $t_i \in I = [0, 1]$ is the local curve parameter. Non-uniform cubic B-spline curves can also be applied to model the object boundary. However, for simplicity, uniform cubic B-spline curves are employed here. Non-uniform splines can be converted to uniform splines through a re-sampling process. A process for computing the b-spline parameters from experimental data can be found in [104].

Suppose that an object is described by a uniform B-spline curve of n segments. If the local parameter intervals are normalized to $[0, 1]$, then a single global parameter u , defined on the interval $I_n \equiv [0, n]$, can be defined to accumulate the values of local curve parameters $\{t_i\}$. The object boundary is usually a closed curve; hence, the two end points of the B-spline curve coincide.

Example 7.7: Fig. 7.5 shows an object modeled by a cubic B-spline curve of eight segments. The global parameter interval is $I_8 = [0, 8]$. The FC-regions in \mathcal{C}_2 are shown in Fig. 7.6. Since the boundary curve is a cubic polynomial, the grasping energy function E is a polynomial of degree 6, as shown in Fig. 7.7 (where $\kappa = 1$). Table 7.1 lists the global maximum and local extrema of E found by TRUST. Since E is symmetric with respect to the line $u_1 = u_2$, only extrema with $u_1 > u_2$ are listed. The corresponding maximal and antipodal point grasp locations are shown in Fig. 7.5. ■

| Extrema | Contact Config. | Antipodal Points | |
|----------------------------|-----------------|------------------|-----------------|
| Global Max (\triangle) | (3.999, 0.071) | (-0.100, -0.900) | (0.020, 0.801) |
| Local Max (\diamond) | (6.070, 2.003) | (-0.704, 0.023) | (0.700, 0.098) |
| Local Min (\star) | (5.245, 1.027) | (-0.407, -0.272) | (0.312, 0.488) |
| Local Min (\square) | (7.005, 2.875) | (-0.398, 0.403) | (0.370, -0.322) |

Table 7.1: Extrema and antipodal points on the planar object

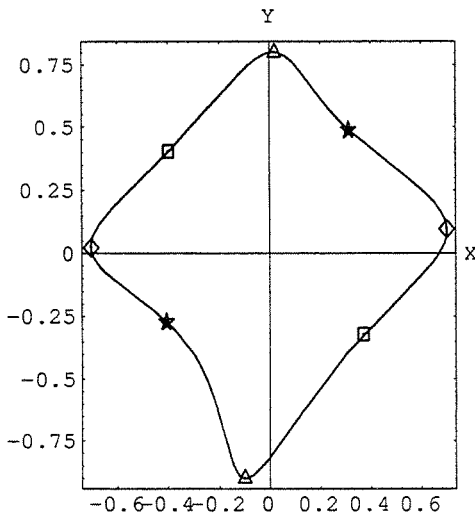


Figure 7.5: A planar object

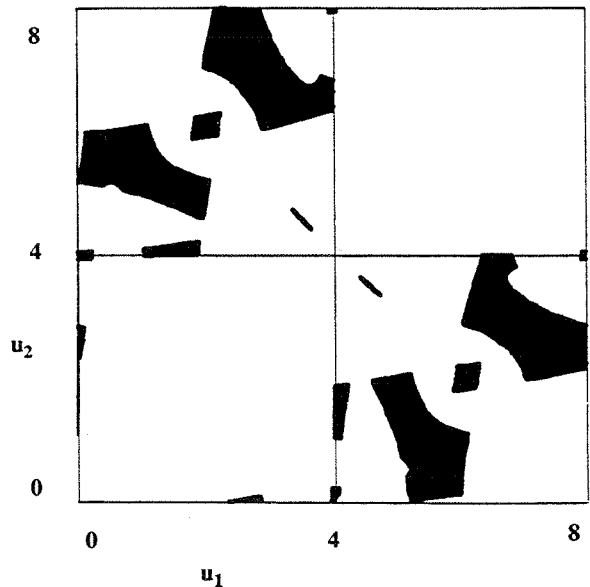


Figure 7.6: FC-region ($\mu = 0.3$)

When part of the object is occluded by nearby obstacles or unviewable by vision sensors, the object boundary can be modeled by one or more open cubic B-spline curves. We can still use the accumulated-value global parameter method for each B-spline curve

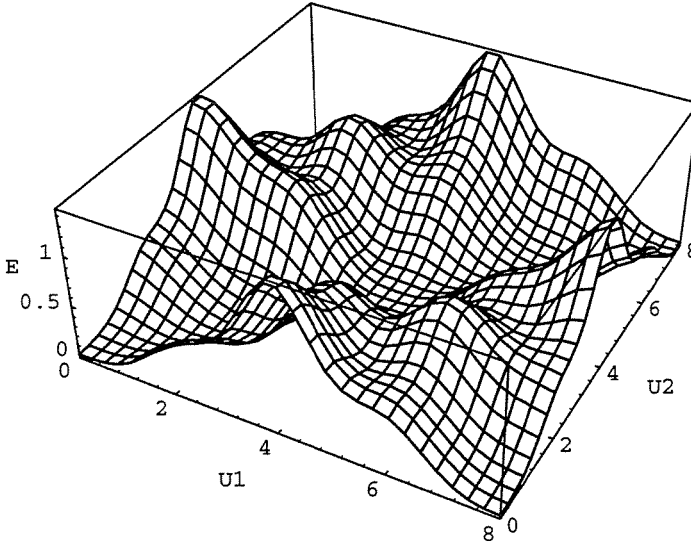


Figure 7.7: Grasping energy function E

and consider all possible combinations of fingertip locations on any of the open curves. The constrained optimization formulation for finding antipodal points still holds.

7.3.2. Spatial Objects

Force-Closure Regions in Contact C-space

One can derive inequalities analogous to (7.6)–(7.9) which define FC-regions in the contact C-space of a 3-D object: $g_1^-(\mathbf{u}_1, \mathbf{u}_2) > 0$, $g_2^-(\mathbf{u}_1, \mathbf{u}_2) > 0$, $g_1^+(\mathbf{u}_1, \mathbf{u}_2) < 0$, and $g_2^+(\mathbf{u}_1, \mathbf{u}_2) < 0$. Let

$$\mathcal{G}^- = \{(\mathbf{u}_1, \mathbf{u}_2) | g_1^- > 0, g_2^- > 0; (\mathbf{u}_1, \mathbf{u}_2) \in \mathcal{D}_2\}$$

$$\mathcal{G}^+ = \{(\mathbf{u}_1, \mathbf{u}_2) | g_1^+ < 0, g_2^+ < 0; (\mathbf{u}_1, \mathbf{u}_2) \in \mathcal{D}_2\},$$

\mathcal{G}^- and \mathcal{G}^+ represent all squeezing and expanding force closure grasps in \mathcal{D}_2 . The two-finger FC-region is the set $\mathcal{G} = \mathcal{G}^- \cup \mathcal{G}^+ \subset \mathcal{D}_2$.

Grasping Energy Function

A 3-D grasping energy function, $E_3: \mathcal{D}_2 \rightarrow \mathbb{R}$, can be defined as

$$E_3(\mathbf{u}_1, \mathbf{u}_2) = \frac{1}{2} \kappa \|\mathbf{s}(\mathbf{u}_1) - \mathbf{s}(\mathbf{u}_2)\|^2. \quad (7.24)$$

Proposition 7.8: A pair of antipodal points, \mathbf{u}_1 and \mathbf{u}_2 , on a 3-D object correspond to a critical point of E_3 . Conversely, critical points of E_3 in the FC-region \mathcal{G} , correspond to antipodal point pairs.

The proof of this proposition is completely analogous to that of Proposition 7.6. However, the optimization of E_3 is difficult to implement because the \mathcal{D}_2 does not admit a global parameterization.

7.3.2.1. Representation of Spatial Objects

Local models for 3-D object surfaces can be developed using techniques, such as B-spline surfaces, Bezier surfaces, etc. [27,104]. The entire object surface is described by a collection of surface patches. Here *spherical product surfaces* are used to globally represent artificial and natural objects. The spherical product was first introduced by Barr [3] to represent a family of parameterized trigonometric 3-D surfaces called *superquadric surfaces*. The basic form of a spherical product surface is defined as [3]

$$\mathbf{s}(u, v) = \mathbf{f}(u) \otimes \mathbf{g}(v) = [f_1(u)g_1(v), f_1(u)g_2(v), f_2(u)] \quad (7.25)$$

where $\mathbf{f}(u)$ and $\mathbf{g}(u)$ are 2-D curves:

$$\mathbf{f}(u) = [f_1(u), f_2(u)], \quad u \in I_u \equiv [u_0, u_1] \quad (7.26)$$

$$\mathbf{g}(v) = [g_1(v), g_2(v)], \quad v \in I_v \equiv [v_0, v_1] \quad (7.27)$$

$\mathbf{f}(u)$ and $\mathbf{g}(v)$ are parametric trigonometric curves in [3]. To represent a richer set of objects, we extend this definition to use B-spline curves instead: let $\mathbf{f} : I_u \equiv I_m \rightarrow \mathbb{R}^2$ be an *open* cubic B-spline curve of m segments and $\mathbf{g} : I_v \equiv I_n \rightarrow \mathbb{R}^2$ a *closed* cubic B-spline curve of n segments. To guarantee object surface smoothness, $\mathbf{f}(u)$ and $\mathbf{g}(v)$ must satisfy the following conditions:

(R-1) $\mathbf{f}(u)$ and $\mathbf{g}(v)$ must be regular curves.

(R-2) The curve $\mathbf{f}(u)$ intersects the y-axis at $\mathbf{f}(0)$ and $\mathbf{f}(m)$ only. The tangents $\mathbf{f}'(0)$ and $\mathbf{f}'(m)$ must have zero slope.

(R-3) The tangent vector and the position vector of a point on $\mathbf{g}(v)$ are not parallel, i.e., $\mathbf{g}(v) \neq \alpha \mathbf{g}'(v)$ for some $\alpha \neq 0$, or $g_1 g_2' - g_1' g_2 \neq 0$.

With the above restrictions, the generalized spherical product surfaces can be used as primitive computer models of real object surfaces [82]. By comparing with range data of real objects, primitive models can be deformed approximately into the shape of real objects via a series of linear and nonlinear transformations such as linear stretching, tapering, or quadratic bending [83]. This is a very versatile object modeling system for real time implementation of object grasping. For simplicity, we investigate grasping on the primitive object form, i.e., spherical product surfaces without distortion, here.

The spherical product, which maps $I_m \times I_n$ onto a surface diffeomorphic to \mathbb{S}^2 , is not a 1-to-1 mapping because the two polar points, $\mathbf{p}_n = \mathbf{s}(0, v)$ and $\mathbf{p}_s = \mathbf{s}(m, v)$, $v \in I_n$, are actually degenerate curves. However, it can be shown [27] that the normal vectors at the polar points are well defined and continuous in the neighborhoods of \mathbf{p}_n and \mathbf{p}_s .

The domain of the spherical product surface is $I_u \times I_v$, so the contact C-space becomes $\mathcal{D}_2^* = I_u \times I_v \times I_u \times I_v$, which is topologically different from \mathcal{D}_2 . A grasp configuration is thus denoted by $\mathbf{q} = (u_1, v_1, u_2, v_2)$. Let $\mathbf{s}_u \equiv \frac{\partial \mathbf{s}}{\partial u}|_{(u_0, v_0)}$ and $\mathbf{s}_v \equiv \frac{\partial \mathbf{s}}{\partial v}|_{(u_0, v_0)}$ denote surface tangent vectors at $\mathbf{p}_0 = \mathbf{s}(u_0, v_0)$ along u and v direction respectively. The unit outward normal vector at \mathbf{p}_0 is

$$\mathbf{n}^+(u_0, v_0) = \pm \frac{\mathbf{s}_u \times \mathbf{s}_v}{\|\mathbf{s}_u \times \mathbf{s}_v\|} \quad (7.28)$$

except at \mathbf{p}_n and \mathbf{p}_s . The sign of \mathbf{n}^+ depends on the directions of parameterization of curves $\mathbf{f}(u)$ and $\mathbf{g}(v)$. The unit outward normal vectors at polar points are $\mathbf{n}_{\mathbf{p}_n}^+ = [0, 0, 1]$ and $\mathbf{n}_{\mathbf{p}_s}^+ = [0, 0, -1]$.

We divide the finger contact space $\mathcal{D}_2^* = \{(u_1, v_1, u_2, v_2) \mid 0 \leq u_i \leq m, 0 \leq v_i \leq n, i =$

$1, 2\}$ into six subsets to determine the FC-regions in \mathcal{D}_2^* :

$$\mathcal{D}_{21}^* = \{\mathbf{u} \mid 0 < u_i < m, 0 \leq v_i \leq n, i = 1, 2\},$$

$$\mathcal{D}_{22}^* = \{\mathbf{u} \mid u_1 = 0, 0 < u_2 < m, 0 \leq v_i \leq n, i = 1, 2\},$$

$$\mathcal{D}_{23}^* = \{\mathbf{u} \mid u_1 = m, 0 < u_2 < m, 0 \leq v_i \leq n, i = 1, 2\},$$

$$\mathcal{D}_{24}^* = \{\mathbf{u} \mid u_2 = 0, 0 < u_1 < m, 0 \leq v_i \leq n, i = 1, 2\},$$

$$\mathcal{D}_{25}^* = \{\mathbf{u} \mid u_2 = m, 0 < u_1 < m, 0 \leq v_i \leq n, i = 1, 2\},$$

$$\mathcal{D}_{26}^* = \{(0, v_1, 0, v_2), (0, v_1, m, v_2), (m, v_1, 0, v_2), (m, v_1, m, v_2) \mid 0 \leq v_i \leq n, i = 1, 2\}.$$

\mathcal{D}_{21}^* represents all two-finger grasps which do not include a polar point. \mathcal{D}_{22}^* and \mathcal{D}_{23}^* represent grasp configurations where finger 1 is located at \mathbf{p}_n or \mathbf{p}_s while finger 2 is located anywhere except at the polar points. \mathcal{D}_{24}^* and \mathcal{D}_{25}^* are similar to \mathcal{D}_{22}^* and \mathcal{D}_{23}^* with finger 1 and finger 2 switching roles. \mathcal{D}_{26}^* represents the four possible contacts of fingers 1 and 2 at \mathbf{p}_n and \mathbf{p}_s . The FC-regions, \mathcal{G}_i^* , in each subset \mathcal{D}_{2i}^* are derived by substituting (7.28), \mathbf{n}_{P_n} , and \mathbf{n}_{P_s} into (7.6) to (7.9). The entire force-closure set in \mathcal{D}_2^* is the union of FC-regions in every subset, i.e., $\mathcal{G}^* = \bigcup_{i=1}^6 \mathcal{G}_i^*$. Antipodal points can be found by finding all critical points of (7.24) lying in \mathcal{G}^* . The constrained critical point finding methods discussed in section 7.3.1.3 are applied to each subset \mathcal{D}_{2i}^* separately.

| Contact Configuration | Antipodal Points | |
|--------------------------|----------------------|-----------------------|
| Global Max (Δ) | | |
| (1.39, 0.07, 2.95, 4.00) | (0.01, 0.48, 0.21) | (-0.07, -0.64, -0.48) |
| Local Max (\circ) | | |
| (1.44, 4.00, 2.98, 0.07) | (-0.06, -0.55, 0.20) | (0.01, 0.56, -0.49) |
| Local Max (\square) | | |
| (1.10, 2.01, 3.16, 6.07) | (0.37, -0.05, -0.27) | (-0.44, -0.01, -0.56) |
| Local Max (+) | | |
| (1.10, 6.07, 3.16, 2.00) | (-0.38, 0.01, 0.28) | (0.44, 0.06, -0.59) |
| Local Max (\star) | | |
| (0.0, 0.0, 4.0, 2.0) | (0.0, 0.0, 0.4) | (0.0, 0.0, -0.7) |

Table 7.2: Extrema and antipodal points on the spatial object

Example 7.9: Fig. 7.8 shows a spherical product surface $\mathbf{f}(u) \otimes \mathbf{g}(v)$. The result of using TRUST in every subset of \mathcal{D}_2^* is listed in Table 7.2. The corresponding antipodal

point grasps are shown in Fig. 7.8 and 7.9 from different view points. Because of symmetry, we list grasps with $u_1 < u_2$ only. Note that all these grasps correspond to grasping energy minima or maxima. Antipodal point grasps corresponding to saddles of E_3 are not listed here because TRUST will “escape” those saddle points during the optimizing process.

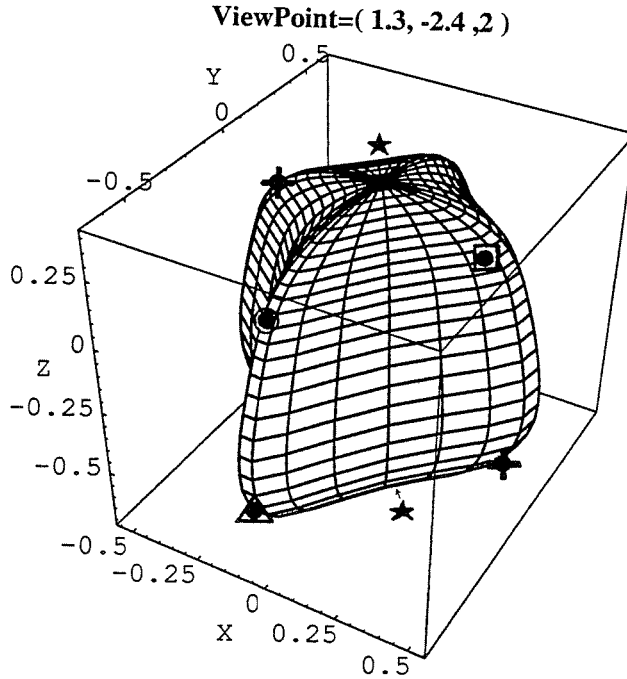


Figure 7.8: A spatial object parameterized by spherical product

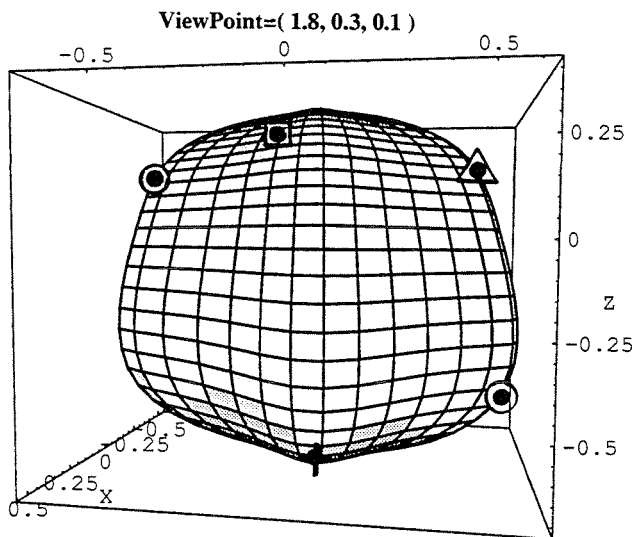


Figure 7.9: From different viewpoint

7.4. N-Finger Grasp Planning

7.4.1. A Qualitative Force-Closure Test Function

From a computational point of view, the third statement of Proposition 7.4 gives us a plausible way to determine if an n-finger planar grasp \mathbf{q} is force-closure. Let \mathbf{n}_{ij} be the vector normal to the plane E_{ij} defined above: $\mathbf{n}_{ij} = \mathbf{w}_i \times \mathbf{w}_j$. If the inner product of \mathbf{n}_{ij} with all other column vectors of $W(\mathbf{q})$ are of the same sign or equal to zero, that means all these vectors are lying on the closed half space defined by E_{ij} . Thus, E_{ij} becomes a supporting plane of $CO(W)$, and force closure is not obtained. For n-finger grasps, the grasp map W has $2n$ column vectors \mathbf{w}_i . Each plane E_{ij} is formed by any two of the \mathbf{w}_i 's, and there are $\binom{2n}{2} = n(2n - 1)$ such planes which must be checked for the supporting hyperplane. If none of the plane E_{ij} is a supporting plane, a force-closure grasp is achieved. According to this statement, we first define a test function which will be TRUE if all of its arguments are of the same sign or equal to zero.

Let $\text{SameSignQ}[x_1, \dots, x_n]$ be a test function of n real arguments x_1, \dots, x_n such that

$$\text{SameSignQ}[x_1, \dots, x_n] = \begin{cases} \text{TRUE}, & \text{if either } \forall i, x_i \geq 0, \text{ or } \forall i, x_i \leq 0; \\ \text{FALSE}, & \text{if } x_i\text{'s are not of the same sign.} \end{cases} \quad (7.29)$$

Based on this test function, we define another test function to determine if the plane E_{ij} is a supporting plane of $CO(W)$:

$$\text{FACE}_{ij}(\mathbf{q}) = \text{SameSignQ}[\mathbf{n}_{ij} \cdot \mathbf{w}_1, \mathbf{n}_{ij} \cdot \mathbf{w}_2, \dots, \mathbf{n}_{ij} \cdot \mathbf{w}_k], \quad k = 1, \dots, 2n, k \neq i, j \quad (7.30)$$

If $\text{FACE}_{ij} = \text{TRUE}$ then the E_{ij} is a supporting plane.

Definition 7.10: The test

$$\text{ForceClosure}(\mathbf{q}) = \neg \left(\bigvee_{\substack{i,j=1 \\ i \neq j, i < j}}^{2n} \text{FACE}_{ij}(\mathbf{q}) \right) \quad (7.31)$$

tells us the force-closure condition of an n-finger grasp \mathbf{q} . $\text{ForceClosure}(\mathbf{q}) = \text{TRUE}$ if and only if \mathbf{q} is force-closure.

The number of arguments in SameSignQ of FACE_{ij} test depends on the number of finger contacts. For n-finger contacts, SameSignQ has $2n - 2$ arguments and there are $\binom{2n}{2}$ FACE_{ij} test functions in ForceClosure.

The situation that an argument in FACE_{ij}, the triple scalar product $\mathbf{n}_{ij} \cdot \mathbf{w}_k = (\mathbf{w}_i \times \mathbf{w}_j) \cdot \mathbf{w}_k$, $i \neq j \neq k$, is equal to zero defines an $(n - 1)$ dimensional hyper surface in the n-contact C-space, \mathcal{C}_n , called a *force-closure surface* (FC-surface) and divides \mathcal{C}_n into two disjoint regions which have different signs of the triple scalar product $(\mathbf{w}_i \times \mathbf{w}_j) \cdot \mathbf{w}_k$.

The number of the FC-surfaces in \mathcal{C}_n is at most $\binom{2n}{2} \cdot (2n - 2)$. However, from the cyclic property of the triple scalar product: $(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = (\mathbf{b} \times \mathbf{c}) \cdot \mathbf{a} = (\mathbf{c} \times \mathbf{a}) \cdot \mathbf{b}$, for $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^3$, the three FC-surfaces $(\mathbf{w}_i \times \mathbf{w}_j) \cdot \mathbf{w}_k = 0$, $(\mathbf{w}_j \times \mathbf{w}_k) \cdot \mathbf{w}_i = 0$, and $(\mathbf{w}_k \times \mathbf{w}_i) \cdot \mathbf{w}_j = 0$ will be identical. Since there are $\binom{2n}{3}$ ways of picking any three \mathbf{w}_i 's from the $2n$ column vectors of W , the number of the FC-surfaces is reduced to $\binom{2n}{3}$. For 2-D objects with symmetry, e.g., a circle, this number may be further reduced.

The FC-surfaces divide the entire \mathcal{C}_n into subregions. Those regions whose contact configurations satisfy ForceClosure(\mathbf{q}) = TRUE are FC-regions. While the number of FC-surfaces can be well characterized, the number of feasible grasping regions cannot. Their number depends on the characteristics of the object boundary curve.

Example 7.11: The above idea will be illustrated with a two-finger grasp. Let \mathbf{f}_i^\pm , $i = 1, 2$ denote the edge vectors of the friction cones; the grasp map W becomes

$$W = \begin{bmatrix} \mathbf{f}_1^+ & \mathbf{f}_1^- & \mathbf{f}_2^+ & \mathbf{f}_2^- \\ \mathbf{p}_1 \otimes \mathbf{f}_1^+ & \mathbf{p}_1 \otimes \mathbf{f}_1^- & \mathbf{p}_2 \otimes \mathbf{f}_2^+ & \mathbf{p}_2 \otimes \mathbf{f}_2^- \end{bmatrix} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \mathbf{w}_3 \quad \mathbf{w}_4] \quad (7.32)$$

Four column vectors of W will form $\binom{4}{2} = 6$ planes E_{ij} passing through the origin of the wrench space; hence there are 6 FACE_{ij}(\mathbf{q}) test functions:

$$\begin{aligned}
\text{FACE}_{12}(\mathbf{q}) &= \text{SameSignQ}[(\mathbf{w}_1 \times \mathbf{w}_2) \cdot \mathbf{w}_3, (\mathbf{w}_1 \times \mathbf{w}_2) \cdot \mathbf{w}_4] \\
\text{FACE}_{13}(\mathbf{q}) &= \text{SameSignQ}[(\mathbf{w}_1 \times \mathbf{w}_3) \cdot \mathbf{w}_2, (\mathbf{w}_1 \times \mathbf{w}_3) \cdot \mathbf{w}_4] \\
\text{FACE}_{14}(\mathbf{q}) &= \text{SameSignQ}[(\mathbf{w}_1 \times \mathbf{w}_4) \cdot \mathbf{w}_2, (\mathbf{w}_1 \times \mathbf{w}_4) \cdot \mathbf{w}_3] \\
\text{FACE}_{23}(\mathbf{q}) &= \text{SameSignQ}[(\mathbf{w}_2 \times \mathbf{w}_3) \cdot \mathbf{w}_1, (\mathbf{w}_2 \times \mathbf{w}_3) \cdot \mathbf{w}_4] \\
\text{FACE}_{24}(\mathbf{q}) &= \text{SameSignQ}[(\mathbf{w}_2 \times \mathbf{w}_4) \cdot \mathbf{w}_1, (\mathbf{w}_2 \times \mathbf{w}_4) \cdot \mathbf{w}_3] \\
\text{FACE}_{34}(\mathbf{q}) &= \text{SameSignQ}[(\mathbf{w}_3 \times \mathbf{w}_4) \cdot \mathbf{w}_1, (\mathbf{w}_3 \times \mathbf{w}_4) \cdot \mathbf{w}_2]
\end{aligned} \tag{7.33}$$

The force-closure test is

$$\text{ForceClosure}(\mathbf{q}) = \neg \left(\bigvee_{\substack{i,j=1 \\ i \neq j, i < j}}^4 \text{FACE}_{ij}(\mathbf{q}) \right) \tag{7.34}$$

By the cyclic property, there are only four different FC-surfaces (FC-curves in \mathcal{C}_2):

$$\begin{aligned}
(\mathbf{w}_1 \times \mathbf{w}_2) \cdot \mathbf{w}_3(\mathbf{q}) &= 0, & (\mathbf{w}_1 \times \mathbf{w}_2) \cdot \mathbf{w}_4(\mathbf{q}) &= 0, \\
(\mathbf{w}_1 \times \mathbf{w}_3) \cdot \mathbf{w}_4(\mathbf{q}) &= 0, & (\mathbf{w}_2 \times \mathbf{w}_3) \cdot \mathbf{w}_4(\mathbf{q}) &= 0,
\end{aligned}$$

Figure 7.10 illustrates the FC-curves and the FC-regions in \mathcal{C}_2 of an ellipse defined by $u \mapsto (4 \cos u, 2.5 \sin u)$, $0 \leq u \leq 2\pi$. The friction coefficient $\mu = 0.3$. \mathcal{C}_2 can be realized by a rectangle $I_{2\pi} \times I_{2\pi}$, ($I_{2\pi} = [0, 2\pi]$), with opposite edges connected.

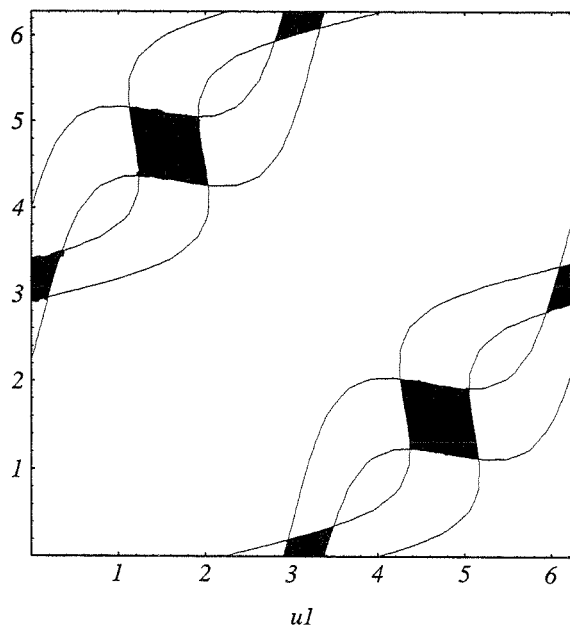


Figure 7.10: FC-curves and FC-regions of an ellipse

It can be shown that this method of computing the force-closure regions based on the qualitative test of equation (7.31) is equivalent to the computation of the force-closure regions using the methods proposed by Nguyen [72] and Chen and Burdick [12]. However, this approach has the advantage that it simply generalizes to n fingers, whereas the method in [72] is specialized for two fingers.

7.4.2. Properties and Symmetries of the FC-Surfaces

FC-surfaces in the contact C-space have similar algebraic structures due to their symmetry with respect to interchange or permutation of the fingers. This section demonstrates that by permuting contact variables u_1, \dots, u_n in a grasp \mathbf{q} , the actual computation of FC-surfaces can be simplified for the n finger case.

The FC-surfaces are formed by equating the triple scalar product of any three of the $2n$ column vectors in $W(\mathbf{q})$ to zero. The three vectors in the triple scalar product may come from either: (1) any two of the n contacts: e.g., $(\mathbf{w}_1 \times \mathbf{w}_2) \cdot \mathbf{w}_3$ is formed by two wrenches from u_1 and one from u_2 , or (2) any three of the n contacts: e.g., $(\mathbf{w}_1 \times \mathbf{w}_3) \cdot \mathbf{w}_5$ is formed by one wrench from u_1 , u_2 , and u_3 . We call the FC-surfaces formed by (1) the Type I FC-surfaces, and those by (2), the Type II FC-surfaces. Their algebraic structures are described as follows.

7.4.2.1. Type I FC-Surfaces

For two contacts, u_i and u_j , the four edge wrenches can form four different triple scalar products, hence four FC-surfaces. The implicit representation of these surfaces are:

$$[\mathbf{w}_{2i-1}(u_i) \times \mathbf{w}_{2i}(u_i)] \cdot \mathbf{w}_{2j-1}(u_j) = 0 \quad (7.35)$$

$$[\mathbf{w}_{2i-1}(u_i) \times \mathbf{w}_{2i}(u_i)] \cdot \mathbf{w}_{2j}(u_j) = 0 \quad (7.36)$$

$$[\mathbf{w}_{2i-1}(u_i) \times \mathbf{w}_{2j-1}(u_j)] \cdot \mathbf{w}_{2j}(u_j) = [\mathbf{w}_{2j-1}(u_j) \times \mathbf{w}_{2j}(u_j)] \cdot \mathbf{w}_{2i-1}(u_i) = 0 \quad (7.37)$$

$$[\mathbf{w}_{2i}(u_i) \times \mathbf{w}_{2j-1}(u_j)] \cdot \mathbf{w}_{2j}(u_j) = [\mathbf{w}_{2j-1}(u_j) \times \mathbf{w}_{2j}(u_j)] \cdot \mathbf{w}_{2i}(u_i) = 0. \quad (7.38)$$

Denote these surfaces by H_{ij}^n since they relate only two contacts, u_i and u_j . The other $n - 2$ contacts are not involved. In (7.13), the edge wrenches $\mathbf{w}_{2k-1}(u_k)$ and $\mathbf{w}_{2k}(u_k)$

are functions of the k -th contact variable u_k only, and $\mathbf{w}_1, \dots, \mathbf{w}_{2k-1}$, (or $\mathbf{w}_2, \dots, \mathbf{w}_{2k}$), $k = 1, \dots, n$, represent positive and negative edge wrenches respectively, so (7.35) and (7.37) (or (7.36) and (7.38)) have similar algebraic structure when they are expanded out, and the variables u_i and u_j are swapped. Hence, we need only to compute (7.35) and (7.36). FC-surfaces (7.37) and (7.38) are obtained by substituting the variables: $\{u_i \rightarrow u_j, u_j \rightarrow u_i\}$ into (7.35) and (7.36). There are $\binom{n}{2}$ ways of selecting any two of the n contacts. Every pair of contacts u_l and u_m generate four FC-surfaces H_{lm}^n similar to (7.35) to (7.38) by substituting the variables: $\{u_i \rightarrow u_l, u_j \rightarrow u_m\}$ into (7.35) to (7.38). In total, there are $4 \cdot \binom{n}{2}$ Type I FC-surfaces in \mathcal{C}_n .

7.4.2.2. Type II FC-Surfaces

For three contacts u_i, u_j , and u_k , the eight edge wrenches can form eight different triple scalar products, hence eight FC-surfaces, denoted by H_{ijk}^n :

$$[\mathbf{w}_{2i-1}(u_i) \times \mathbf{w}_{2j-1}(u_j)] \cdot \mathbf{w}_{2k-1}(u_k) = 0 \quad (7.39)$$

$$[\mathbf{w}_{2i-1}(u_i) \times \mathbf{w}_{2j-1}(u_j)] \cdot \mathbf{w}_{2k}(u_k) = 0 \quad (7.40)$$

$$[\mathbf{w}_{2i-1}(u_i) \times \mathbf{w}_{2j}(u_j)] \cdot \mathbf{w}_{2k-1}(u_k) = [\mathbf{w}_{2k-1}(u_k) \times \mathbf{w}_{2i-1}(u_i)] \cdot \mathbf{w}_{2j}(u_j) = 0 \quad (7.41)$$

$$[\mathbf{w}_{2i-1}(u_i) \times \mathbf{w}_{2j}(u_j)] \cdot \mathbf{w}_{2k}(u_k) = 0 \quad (7.42)$$

$$[\mathbf{w}_{2i}(u_i) \times \mathbf{w}_{2j-1}(u_j)] \cdot \mathbf{w}_{2k-1}(u_k) = [\mathbf{w}_{2j-1}(u_j) \times \mathbf{w}_{2k-1}(u_k)] \cdot \mathbf{w}_{2i}(u_i) = 0 \quad (7.43)$$

$$[\mathbf{w}_{2i}(u_i) \times \mathbf{w}_{2j-1}(u_j)] \cdot \mathbf{w}_{2k}(u_k) = [\mathbf{w}_{2j-1}(u_j) \times \mathbf{w}_{2k}(u_k)] \cdot \mathbf{w}_{2i}(u_i) = 0 \quad (7.44)$$

$$[\mathbf{w}_{2i}(u_i) \times \mathbf{w}_{2j}(u_j)] \cdot \mathbf{w}_{2k-1}(u_k) = [\mathbf{w}_{2k-1}(u_k) \times \mathbf{w}_{2i}(u_i)] \cdot \mathbf{w}_{2j}(u_j) = 0 \quad (7.45)$$

$$[\mathbf{w}_{2i}(u_i) \times \mathbf{w}_{2j}(u_j)] \cdot \mathbf{w}_{2k}(u_k) = 0 \quad (7.46)$$

Observe that the similarity between (7.40), (7.41), and (7.43) (or (7.42), (7.44), and (7.45)). In this case, we need only compute (7.39), (7.40), (7.42), and (7.46). By substituting the variables: $\{u_i \rightarrow u_k, u_j \rightarrow u_i, u_k \rightarrow u_j\}$ and $\{u_i \rightarrow u_j, u_j \rightarrow u_k, u_k \rightarrow u_i\}$ in (7.40) and (7.42), we can obtain (7.41) and (7.45), (7.43) and (7.44) respectively. There are $\binom{n}{3}$ ways of selecting any three of the n contacts. The three contacts u_p, u_q , and u_r generate eight FC-surfaces H_{pqr}^n by substituting the variables: $\{u_i \rightarrow u_p, u_j \rightarrow u_q, u_k \rightarrow u_r\}$ into (7.39) to (7.46). Consequently, there are $8 \cdot \binom{n}{3}$ Type II FC-surfaces.

Note that the total number of Type I and Type II FC-surfaces is:

$$4\binom{n}{2} + 8\binom{n}{3} = \binom{2n}{3} \quad n \geq 3. \quad (7.47)$$

This number is equal to the number of FC-surfaces mentioned in Section 7.4.1. This implies that all FC-surfaces in an n -contact C-space are either Type I or II FC-surfaces. *The number of the FC-surfaces equations which must actually be computed is six, i.e., equation (7.35), (7.36), (7.39), (7.40), (7.42), and (7.46).* The rest of the equations can be found by substituting different variables into the above six equations. Thus, the computation of FC-surfaces can be highly simplified.

7.4.3. Force-Closure Contact Modes

This section considers a characterization of the force-closure sets in terms of the number of fingers necessary to effect force-closure. This characterization and decomposition of the sets is essential to the implementation of finger gaits.

7.4.3.1. Definition

Assuming point contact with friction, at least two fingers are required to implement a force-closure grasp on a planar object. These two contacts are called an *FC-2 contact*, as the contacts are arranged in such a way that only two fingers are required to guarantee closure.

However, a three-finger grasp can be force-closure in one of two ways: 1) any two of the three finger contacts form force-closure, or 2) three of the contacts satisfy the closure condition but no two contacts satisfy force-closure. The latter type of force-closure contact configuration is called an *FC-3 contact*. In general, an *FC- n contact* is formed by n contacts whose $2n$ edge wrenches positively span the wrench space \mathbb{R}^3 and no subset of k ($2 \leq k \leq n - 1$) fingers forms a force-closure grasp.

Definition 7.12: An n -finger grasp \mathbf{q} is called an FC- m contact grasp ($m \leq n$) if m of the n finger contacts form an FC- m contact.

Hence, only FC-2 and FC-3 contact modes are available for three-finger FC-grasps. But in a four-finger FC-grasp, FC-2, FC-3, FC-4, or the combination of FC-2 and FC-3 contact modes are all possible. (In the last case, FC-2 and FC-3 contacts share a common finger contact.) However, the number of FC-contact modes has an upper bound due to the following theorem from convex analysis.

Theorem: (Steinitz)[20,65]

Let $X \subset \mathbb{R}^n$ be a finite set, i.e., $X = \{x_1, \dots, x_m\}$, $x_i \in \mathbb{R}^n$, $i = 1, \dots, m$ and $0 \in \text{Interior}(CO(X))$; then there exists a $Y \subseteq X$ such that $0 \in \text{Interior}(CO(Y))$ with $|Y| \leq 2n$.

Proposition 7.13: Assuming point contact with friction, there exists at most FC-6 contact modes in an n -finger FC-grasp on a planar object for $n \geq 6$.

Proof: From Proposition 7.4, we know that in an n -finger force-closure grasp \mathbf{q} , the $2n$ column vectors (or edge wrenches) of the grasp map $W(\mathbf{q})$ positively span the wrench space \mathbb{R}^3 or, equivalently, $0 \in \text{Interior}(CO(W))$. But from Steinitz theorem, in this case, $n = 3$; at most six of the $2n$ edge wrenches, called $\{w_{k1}, \dots, w_{k6}\}$, are needed to positively span \mathbb{R}^3 , or $0 \in \text{Interior}(CO(\{w_{k1}, \dots, w_{k6}\}))$. Since each finger contact generates two edge wrenches and the number of finger contacts is equal to or greater than six, in the worst case, the six wrenches $\{w_{k1}, \dots, w_{k6}\}$ come from one of the two edge wrenches of each of the six contacts. Under this circumstance, six finger contacts are required for force-closure. Any five or less of the n finger contacts may not form force-closure. ■

This proposition has the following physical interpretation. If a p -finger ($p \geq 7$) planar grasp is force-closure, then there *must* exist a subset of six fingers which is also force-closure. This does not imply that any subset of six fingers will be force-closure, but only that at least one choice of six fingers will be force-closure. Consequently, it is always possible to lift $(p - 6)$ fingers from the object surface such that the force-closure condition on the object is not disturbed. Thus, in some respects, it is relatively easy to

plan finger gaits for planar grasps with seven or greater fingers. From a practical point of view, it is not desirable to build seven-fingered grippers solely to simplify finger gait planning.

The same is not true for less than seven-fingered planar grasps. In these cases, for a given n -fingered ($n \leq 6$) force-closure grasp, it may not be possible to lift a finger so that force-closure is maintained. If $n \geq 3$, then there do exist force-closure grasps in which d ($1 \leq d \leq n - 2$) fingers can be lifted while maintaining force-closure. These force-closure grasps are a subset of all force-closure grasps. Hence, we need an effective procedure for identifying the FC- m subsets of closure-grasps in order to implement finger gaiting.

7.4.3.2. Identifying FC-Contact Modes in a Grasp

In this section, we introduce an algorithm to determine what FC-contact modes and the corresponding ones a force-closure grasp \mathbf{q} may have. The set notation $\mathbf{q}_n = \{u_1, \dots, u_n\}$ and the index set of contacts $I = \{1, \dots, n\}$ are used instead of the ordered n -tuple $\mathbf{q} = (u_1, \dots, u_n) \in \mathcal{C}_n$ as an n -finger grasp in this section. All m -contact ($2 \leq m \leq n - 1$) ForceClosure test functions are employed in this algorithm. And let $\text{FC}_m(\mathbf{q}_k) \equiv \text{ForceClosure}(\mathbf{q})$, $\mathbf{q} \in \mathcal{C}_k$.

The algorithm is stated as follow:

```

0  Procedure FCGRASPID( $\mathbf{q}_n$ )
1   $FCList = \emptyset$ ;
2  while  $\text{FC}_n(\mathbf{q}_n) = \text{TRUE}$  do
3      {
4       $k = 2$ ;
5      while  $k < n$  do
6          {
7          let  $Q_k = \{\mathbf{q}_k = \{u_{p_1}, \dots, u_{p_k}\}, p_i \in I, p_i \neq p_j \mid \mathbf{q}_{fc} \not\subset \mathbf{q}_k, \forall \mathbf{q}_{fc} \in FCList\}$ ;
8          for all  $\mathbf{q}_k \in Q_k$  do
9              {
10             if  $\text{FC}_k(\mathbf{q}_k) = \text{TRUE}$  then add  $\mathbf{q}_k$  to  $FCList$ 
11             }
12              $k = k + 1$ ;
13         }
14     if  $FCList = \emptyset$  then add  $\mathbf{q}_n$  to  $FCList$ ;

```

```

15     }
16   return FCList;

```

The input to FCGRASPID is the contact configuration of an n -finger grasp \mathbf{q}_n . The output is a set *FCList*, which stores all contacts in \mathbf{q}_n that form FC-contacts. A member of *FCList* is of the form: $\mathbf{q}_{fc} = \{u_{p_1}, \dots, u_{p_k}\}$, $p_i \in I$, $p_i \neq p_j$, $k \leq n$, which indicates that the k contacts u_{p_1}, \dots, u_{p_k} in \mathbf{q}_n form an FC- k contact. The procedure first initializes *FCList*. If \mathbf{q}_n is force-closure, then it starts to search for the FC- k contacts in \mathbf{q}_n . Line 7 generates all possible k contacts from \mathbf{q}_n denoted by $\mathbf{q}_k = \{u_{p_1}, \dots, u_{p_k}\}$, where $p_i \in I$, excludes those who contain lower FC-contact modes, i.e., FC-2, \dots , FC- $(n-1)$ contacts, and stores them in Q_k . Lines 8-11 check the force-closure condition of all \mathbf{q}_k 's in Q_k . If a \mathbf{q}_k is force-closure, then store it in *FCList*. After checking all \mathbf{q}_k 's, the process then goes back to line 5 and repeats line 5 to line 13 to search for FC- $(k+1)$ contact modes in \mathbf{q}_n until k is equal to $n-1$. If *FCList* is still empty after checking all FC- $(n-1)$, \dots , FC-2 contacts, \mathbf{q}_n is then added to *FCList*. The grasp is an FC- n contact grasp. Line 16 returns *FCList*. If \mathbf{q}_n is not force-closure, *FCList* = \emptyset . If it is, *FCList* gives all FC-contacts in it.

7.4.4. Characterization of the n -finger Force-Closure Sets

A force-closure set (FC-set) is defined to be a set of points which satisfy closure. An FC-region is a connected subset of an FC-set. The characteristics of FC-sets in \mathcal{C}_2 have previously been studied in [11,28]. In this case, there is only a single FC-contact mode: the FC-2 mode. For grasps with more than two contacts, there exists more than one FC-contact mode. The characteristics of the FC-sets and FC-regions in \mathcal{C}_n ($n \geq 3$) will be different from those in \mathcal{C}_2 . A three-finger grasp example is used to illustrate the characteristics of the FC-sets in the higher dimensional contact C-space.

The notation $\mathcal{F}_{p_1 \dots p_k}^n$, ($p_i \in I$), is employed to represent the FC-sets in \mathcal{C}_n in which the p_1, \dots, p_k contacts of all n -finger grasps form FC- k contacts. (Note that $k \leq n$ when $n \leq 6$, and $k \leq 6$ for $n > 6$.) For instance, all two-finger force-closure grasps belong to \mathcal{F}_{12}^2 in \mathcal{C}_2 . As shown in Example 7.11, all shaded areas are \mathcal{F}_{12}^2 .

Generically, the FC-regions in \mathcal{C}_3 are divided into the following: 1) FC-2 contact sets: $\mathcal{F}_{12}^3, \mathcal{F}_{23}^3, \mathcal{F}_{31}^3$, and 2) FC-3 contact sets: \mathcal{F}_{123}^3 . In general, for $n \leq 6$ and $2 \leq m < n$ there will be $\binom{n}{m}$ FC- m contact sets, each arising from a given choice of m fingers. For $n > 6$, there are at most $\binom{n}{6}$ FC-6 sets.

Example 7.14: Consider a three-finger grasp of a circular disk with radius r . The boundary of the disk is described by $\theta \mapsto (r \cos \theta, r \sin \theta)$, $0 \leq \theta \leq 2\pi$. The friction coefficient is assumed to be $\mu = 0.3$. A contact configuration is a triplet $\mathbf{q} = (\theta_1, \theta_2, \theta_3)$, $\theta_i \in \mathbb{S}^1, i = 1, 2, 3$, and $\theta_1 \neq \theta_2 \neq \theta_3$. By Definition 7.2, the three-contact C-space is $\mathcal{C}_3 = \mathbb{T}^3 \setminus (\Delta_{12} \cup \Delta_{23} \cup \Delta_{31})$; it can be realized by a cube $I_{2\pi} \times I_{2\pi} \times I_{2\pi} \in \mathbb{R}^3$, (where $I_{2\pi} = [0, 2\pi]$), with their opposite faces identified. The grasp map is:

$$W(\mathbf{q}) = r \begin{pmatrix} -c_1 - \mu s_1 & -c_1 + \mu s_1 & -c_2 - \mu s_2 & -c_2 + \mu s_2 & -c_3 - \mu s_3 & -c_3 + \mu s_3 \\ \mu c_1 - s_1 & -\mu c_1 - s_1 & \mu c_2 - s_2 & -\mu c_2 - s_2 & \mu c_3 - s_3 & -\mu c_3 - s_3 \\ r\mu & -r\mu & r\mu & -r\mu & r\mu & -r\mu \end{pmatrix} \quad (7.48)$$

where $c_i = \cos \theta_i$ and $s_i = \sin \theta_i$, $i = 1, 2, 3$. Owing to the circular symmetry of the disk, there are only six FC-surfaces in \mathcal{C}_3 and they are all of Type I:

$$H_{12}^3 : \quad \theta_1 - \theta_2 = 2 \tan^{-1}(\pm \frac{1}{\mu}) \quad (7.49)$$

$$H_{23}^3 : \quad \theta_2 - \theta_3 = 2 \tan^{-1}(\pm \frac{1}{\mu}) \quad (7.50)$$

$$H_{31}^3 : \quad \theta_3 - \theta_1 = 2 \tan^{-1}(\pm \frac{1}{\mu}) \quad (7.51)$$

The Type II FC-surfaces degenerate to the Type I surfaces. When \mathcal{C}_3 is identified with $I_{2\pi}^3$, those FC-surfaces become planes in $I_{2\pi}^3$ as shown in Figure 7.11. Figure 7.12 shows slices of \mathcal{C}_3 for fixed values of θ_3 . Strips (1), (2), and (3) belong to FC-2 sets, \mathcal{F}_{12}^3 , \mathcal{F}_{13}^3 , and \mathcal{F}_{23}^3 respectively. Regions (4) and (5) belong to the FC-3 set, \mathcal{F}_{123}^3 . ■

From the above example, we can see some features of the FC-regions:

1. All FC-2 regions $\mathcal{F}_{p_1 p_2}^3$ are bounded by Type I FC-surfaces $H_{p_1 p_2}^3$. In those regions, only two contacts p_1 and p_2 are needed to maintain force-closure. The third finger contact can be placed anywhere on the object.

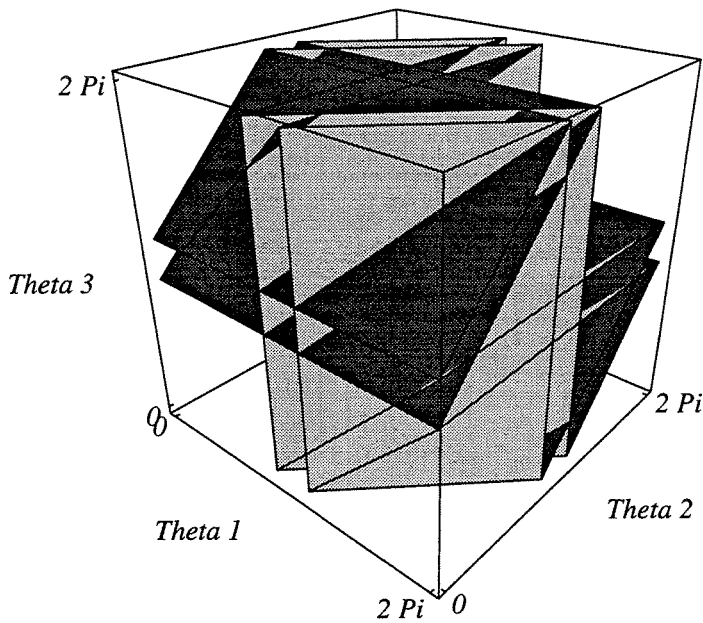


Figure 7.11: FC-surfaces for disk example

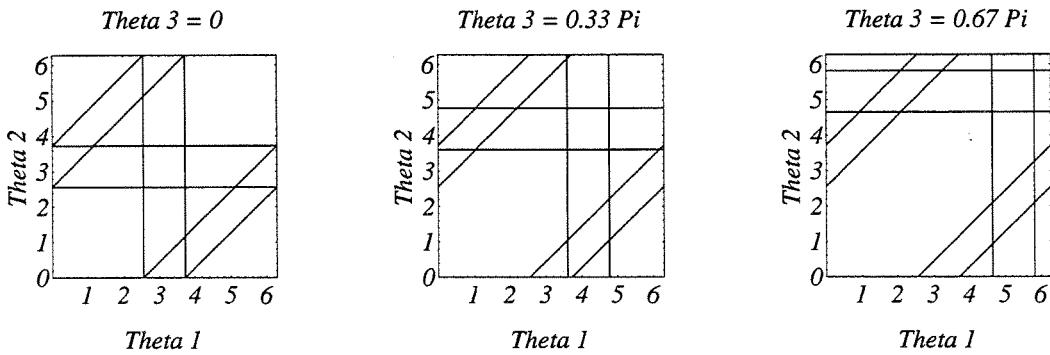


Figure 7.12: Constant θ_3 slices of C_3

2. The FC-3 contact regions \mathcal{F}_{123}^3 are bounded by both Type I and II FC-surfaces, which is also true for higher dimensional contact C-space. Those FC-surface patches that bound the FC-3 regions have to be found by computation. In those regions, all three contacts are required for maintaining force-closure grasp.
3. As seen in the θ_3 -slices of C_3 , there are overlapping regions for FC-2 contact

regions $\mathcal{F}_{p_1 p_2}^3$ and $\mathcal{F}_{p_2 p_3}^3$, i.e., $\mathcal{F}_{p_1 p_2}^3 \cap \mathcal{F}_{p_2 p_3}^3 \neq \emptyset$. For a three-finger force-closure grasp $\mathbf{q} \in \mathcal{F}_{p_1 p_2}^3 \cap \mathcal{F}_{p_2 p_3}^3$, both contact p_1, p_2 and contact p_2, p_3 are FC-2 contacts; thus either contact p_1 or p_3 can break contact with the object without disturbing the force-closure condition.

Definition: The intersection of different FC- m ($m < n$) contact regions are termed *m-finger gait transition regions*.

In this example, the two overlapping FC-2 regions implies that two different pairs of fingers in a 3-finger grasp are by themselves force-closure. In such a region, it is possible to put down one finger (e.g., p_1) and subsequently lift another (e.g., p_3) while maintaining force-closure in all states of the finger repositioning. These regions are essential to the implementation of finger gaits.

Note that $\mathcal{F}_{p_1 p_2}^3 \cap \mathcal{F}_{p_1 p_2}^4 = \emptyset$. For a four-finger contact, any two FC-2 contact regions $\mathcal{F}_{p_1 p_2}^4$ and $\mathcal{F}_{p_3 p_4}^4$ intersect, where $p_i \in I$ and $p_1 \neq p_2, p_3 \neq p_4$. One FC-2 region $\mathcal{F}_{p_1 p_2}^4$ may intersect with one FC-3 region $\mathcal{F}_{p_2 p_3 p_4}^4$. In the intersecting region, the FC-2 contacts and FC-3 contacts share one common contact point p_2 . It is impossible for FC-2 contacts and FC-3 contacts to share two contact points in one grasp. Two FC-3 regions $\mathcal{F}_{p_1 p_2 p_3}^4$ and $\mathcal{F}_{p_2 p_3 p_4}^4$ also intersect. In their intersecting region, the two FC-3 contacts share contacts p_2 and p_3 . By definition, $\mathcal{F}_{1234}^4 \cap \mathcal{F}_{p_1 p_2 p_3}^4 = \emptyset$ and $\mathcal{F}_{1234}^4 \cap \mathcal{F}_{p_1 p_2}^4 = \emptyset$. In contact C-space of dimension greater than 4, the intersection of FC-regions becomes more complicated.

4. The size of the FC-regions depends on the friction coefficient μ . As one can see in the circle example, as μ increase, the FC-2 contact regions grow while the FC-3 contact regions shrink. The total size of the feasible three-finger grasp regions, i.e., both FC-2 sets and FC-3 sets, increases with increasing μ . But the subset which is exclusively two-finger force-closure increases at a faster rate. For contact friction sufficiently large, FC-3 regions disappear (Figure 7.13). This coincides with real world experience that fewer finger contacts are sufficient to maintain a force-closure grasp on an object with large friction contact. For more finger contacts, i.e., in higher dimensional contact C-space, this phenomena also exists.

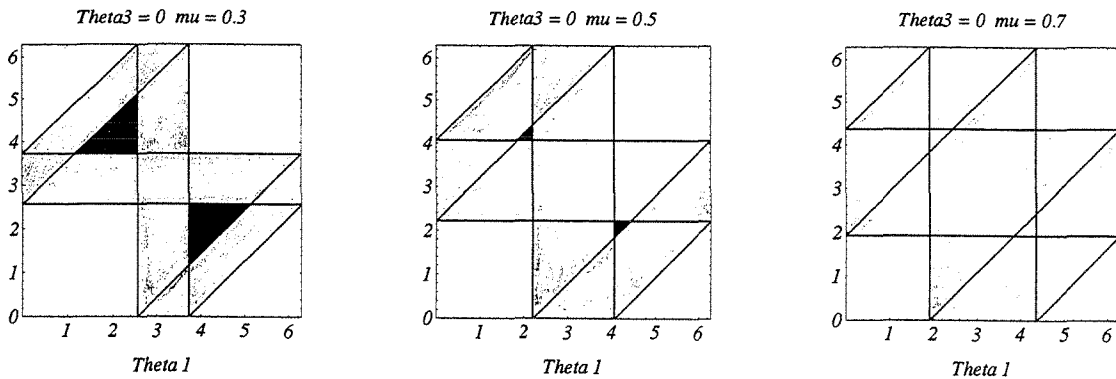


Figure 7.13: $\theta_3 = 0$ slices of C_μ with variable μ

7.4.5. Application to Complex Multifinger Manipulation

This section discusses how to apply the results of the previous sections to planning complex manipulation tasks. Consider the planar multifinger system shown in Fig. 7.14. Each finger has three revolute joints, and all fingers are assumed to be identical. The fingertips are modeled as 2-D objects instead of points. Let F_i be a coordinate frame which is rigidly attached to the i^{th} finger. The boundary curve of each finger tip is described by the curve $c_f : I_r \rightarrow \mathbb{R}^2$ with respect to F_i . As before, we assume that the object to be manipulated is described by a parametric curve, $\mathbf{p}(u)$.

Complex multi-finger manipulation can reposition the grasped object using rolling, sliding, finger repositioning, or any combination of these. The following subsections show how the contact configuration space concept and its decomposition into force closure sets with different order contact modes can be used as a tool for complex manipulation planning. These concepts are demonstrated by a three-finger manipulation of an elliptical object which employs both rolling and gaiting sequences.

7.4.5.1. Multifinger Manipulation

During the manipulating process, the contact locations are moving both on the finger and object surfaces. The evolution of contact points on the finger and object surfaces during the relative motion of these objects is governed by a set of equations called the *contact equations* [57,66,70]. Let u_{oi} denote the contact location on the object by

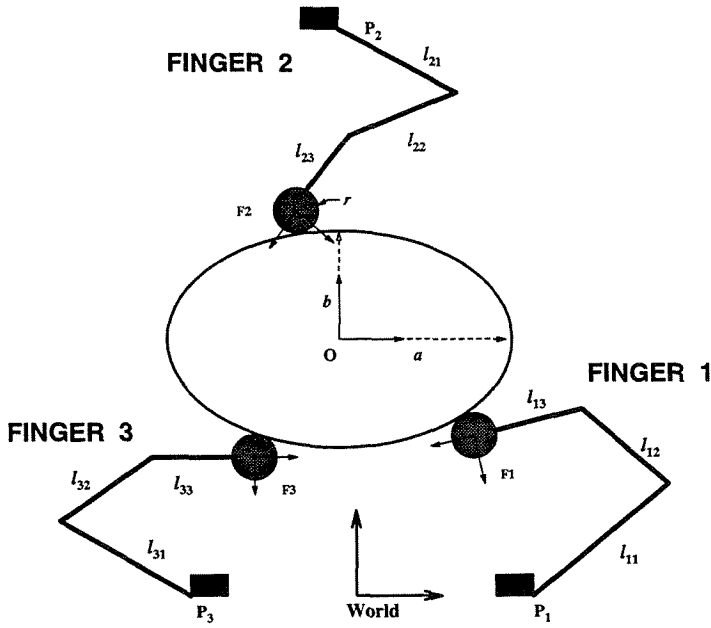


Figure 7.14: A planar three-finger system

i^{th} finger and u_{fi} , the corresponding contact location on the i^{th} fingertip. The planar contact equations are [66]:

$$\dot{u}_{oi} = (k_o + k_{fi})^{-1} M_o^{-1} (\dot{\theta}_i + k_{fi} v_{ti}) \quad (7.52)$$

$$\dot{u}_{fi} = (k_o + k_{fi})^{-1} M_{fi}^{-1} (-\dot{\theta}_i + k_o v_{ti}) \quad (7.53)$$

where k_o and k_{fi} are the curvatures of the object and the finger i at the contact point respectively. $M_o = \|\frac{\partial \mathbf{p}}{\partial u_o}\|$ and $M_{fi} = \|\frac{\partial c_{fi}}{\partial u_{fi}}\|$ are the magnitudes of the tangent vectors at the contact point. If the objects are arclength parameterized, then $M_o = M_{fi} = 1$. $\dot{\theta}_i$ is the relative angular velocity of the object frame O with respect to the finger frame F_i . v_{ti} is the relative linear velocity of O with respect to F_i along the tangent direction at the contact point respectively, e.g., the sliding velocity.

Obviously, the contact configuration, $\mathbf{q}(t) = (u_{o1}(t), u_{o2}(t), u_{o3}(t))$, and the grasp map, $W(\mathbf{q}(t))$, are no longer fixed during a roll/slide manipulation. The evolution of the contact configuration on the object traces out a connected curve segment in the contact C-space during a continuous rolling and sliding sequence. In order to accommodate disturbance forces which arise during the execution of manipulation, force-closure must be maintained at points along the trajectory, $\mathbf{q}(t)$. Equivalently, the entire contact

configuration curve segment, $\mathbf{q}(t)$ $t \in [t_0, t_1]$, must lie in an FC-region in the contact C-space.

This constraint can be accommodated in two ways. In a real time situation, this can be accomplished by setting up the ForceClosure test function and checking at every instance t if $\text{ForceClosure}(\mathbf{q}(t)) = \text{TRUE}$. The ForceClosure test depends on the object geometry and friction coefficient. If these are known in advance, then for a particular object this test function needs to be set up only once.

Alternatively, if the object geometry and friction coefficient are known in advance, then the approach described in this paper can be used off line to plan roll slide motions. Assume an initial contact configuration, \mathbf{q}_0 , is given. Also assume that a final desired contact configuration, \mathbf{q}_f , is also specified. Presumably, both \mathbf{q}_0 and \mathbf{q}_f lie in force-closure regions. To plan a roll/slide manipulation, one must then find a trajectory, $\gamma(t) \in \mathcal{C}_n$, such that $\gamma(t_0) = \mathbf{q}_0$ and $\gamma(t_f) = \mathbf{q}_f$. There may be no feasible trajectory if \mathbf{q}_f does not lie in a force-closure region which is connected to \mathbf{q}_0 . If no trajectory exists, then a finger repositioning event *must* be used, possibly along with a roll/slide motion, to achieve \mathbf{q}_f .

Let us assume that \mathbf{q}_f is in the same connected component of the FC-set as \mathbf{q}_0 , or that \mathbf{q}_f is the final configuration in a continuous motion before a finger repositioning event. If a trajectory $\mathbf{q}(t)$ is chosen, then the $\{\dot{u}_{oi}\}$ are known in (7.52). If we assume that there is no sliding, (7.52) can be uniquely solved for the $\{\dot{\theta}_i\}$ at each instant of time along the trajectory. Otherwise, some combination of rolling and sliding can be chosen to satisfy (7.52). Then, $\dot{\theta}_i$ and v_{ti} from (7.52) can be substituted into (7.53) to determine the \dot{u}_{fi} . The $\{\dot{u}_{fi}\}$, $\{\dot{\theta}_i\}$, and $\{v_{ti}\}$ can then be used, in conjunction with the kinematics of each finger, to determine the finger joint trajectories which implement the desired motion. One must ensure that the proper internal forces are applied as well. This problem has been considered in [42].

The existence of a continuous trajectory in \mathcal{C}_n which connects a starting and final configuration is not sufficient to guarantee that such a manipulation is possible. Large

object displacements can typically not be generated purely by roll/slide manipulations because of finger joint limits, finger surface area limits, and interference between the fingers or fingers and object. Thus, in a most realistic case, finger gaiting will be required in addition to roll/slide motions.

7.4.5.2. Finger Gaits

Here we loosely interpret a finger gait as a sequence of force-closure grasps on the object which maintains a stationary configuration relative to the world frame while also maintaining the force-closure condition during all instances of finger relocation. Finger gaits are employed to lift those fingers that have reached their joint angle or surface area limits, or are about to lose force-closure. By readjusting their postures and relocating them on the object, the next phase of manipulation can be continued. Since a gait usually involves relocating at least two different fingers in sequence, the gait grasps must incorporate at least two different FC-contact modes. Only grasps in the gait transition regions (i.e., the intersection of different FC-contact regions), have more than one FC-contact mode. Thus, complex object manipulations which include finger gaits *must* pass through a transitory state in a gait transition region. Below we give an example which combines finger gaits with rolling manipulation.

7.4.5.3. Dextrous Manipulation Example

Consider using the system in Fig. 7.14 to rotate the ellipse of Example 7.11 by 120° relative to its initial orientation. All fingertips are assumed to be circles of radius $r = 0.7$, and there are no joint limits on the finger joints. All links can rotate 360 degree relative to their neighboring links. However, we do wish to avoid interference between the fingers and the object during the manipulation. This particular task can be accomplished by a sequence of five finger rolling and four finger gaiting motions. Fig. 7.16 shows snapshots from a computer simulation of this complex manipulation employing the contact equations and the FC-test algorithms of the previous sections. Initially, finger 1 and 2 form an FC-2 contact in frame 1. Frames 3 and 6 show the first

rolling stage, in which finger 1, 2 remain FC-2 contact. The rolling motion is stopped at frame 6 because of the impending interference of finger 1 and 2 with the object. The grasp is now in \mathcal{F}_{12}^3 . To reposition finger 1, finger 3 is repositioned in the grasp transition region $\mathcal{F}_{12}^3 \cap \mathcal{F}_{23}^3$ so that finger 2 and 3 also form force-closure in frame 7. After adjusting the posture of finger 1 and putting it in $\mathcal{F}_{12}^3 \cap \mathcal{F}_{23}^3$ (frame 8), finger 3 is relocated in $\mathcal{F}_{12}^3 \cap \mathcal{F}_{13}^3$ to permit release of finger 2 (frame 9). Finger 2 is then readjusted and put in $\mathcal{F}_{12}^3 \cap \mathcal{F}_{13}^3$ (frame 10). Again we relocate finger 3 to an appropriate place in \mathcal{F}_{12}^3 to start manipulating the object in frame 11. By alternating the manipulating and gaiting sequence, we can obtain the desired change in object orientation shown in frame 40.

In this example, all grasps are located in FC-2 contact regions so that one finger can be lifted and put down in another location. Fig. 7.15 shows part of the FC-2 contact regions and the evolution of the contact points in \mathcal{C}_3 of the ellipse. The rolling motions trace out curve segments which lie in FC-2 contact regions \mathcal{F}_{12}^3 and \mathcal{F}_{13}^3 . The gaits are represented by discrete points lying in the gait transition regions $\mathcal{F}_{12}^3 \cap \mathcal{F}_{13}^3$, $\mathcal{F}_{12}^3 \cap \mathcal{F}_{23}^3$, and $\mathcal{F}_{13}^3 \cap \mathcal{F}_{23}^3$. Note that the available force-closure grasps in the gait transition regions are restricted by the interference between fingertips. In our example, the placement of fingertips in a finger gait is chosen manually.

7.4.5.4. Dexterous Manipulation with Sliding

Note that only rolling motion is considered in the above manipulation. Sliding manipulation can be difficult to implement in practice, as it requires explicit knowledge of the object and finger friction coefficient. Also, the friction between two objects becomes dynamic friction during sliding, which is usually different from static friction. Our basic assumption on the static friction contact between fingers and the object is no longer valid. Nevertheless, the methods outlined in this section can be useful for planning sliding motion as well. Consider a three-finger planar grasp. One could plan a robust sliding motion in an FC-2 contact region so that the two non-sliding fingers form a force-closure grasp. The object surface is used to “guide” the motion of the

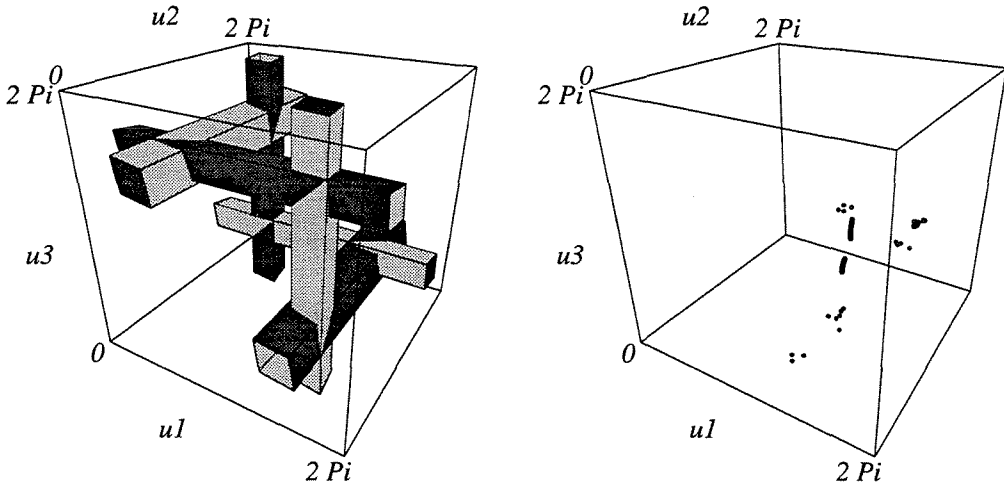


Figure 7.15: FC-2 contact regions and the trajectories of the contact points on the ellipse

sliding finger. The force and moment caused by sliding motion can be treated like disturbance on the object and can be balanced by fingers that form force-closure (Figure 7.17). This appears to be the strategy used by humans during such manipulations as twirling a pencil with three fingers.

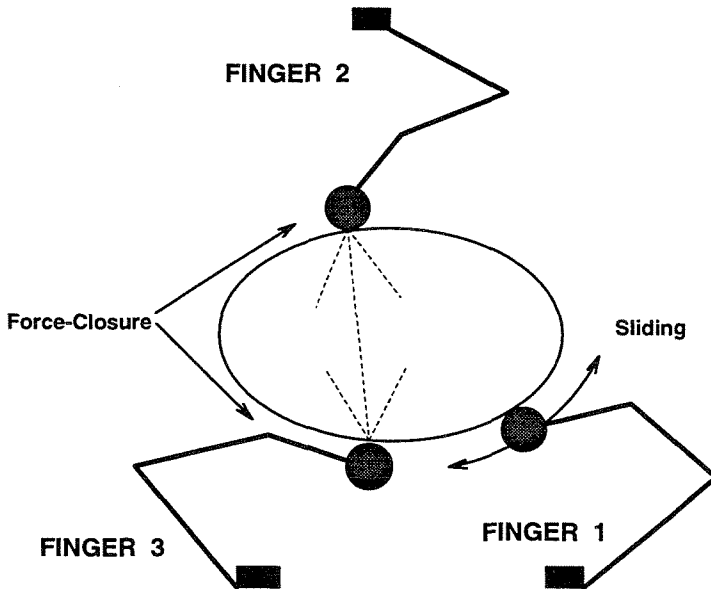


Figure 7.17: Finger 1 slides while Fingers 2 and 3 are force-closure

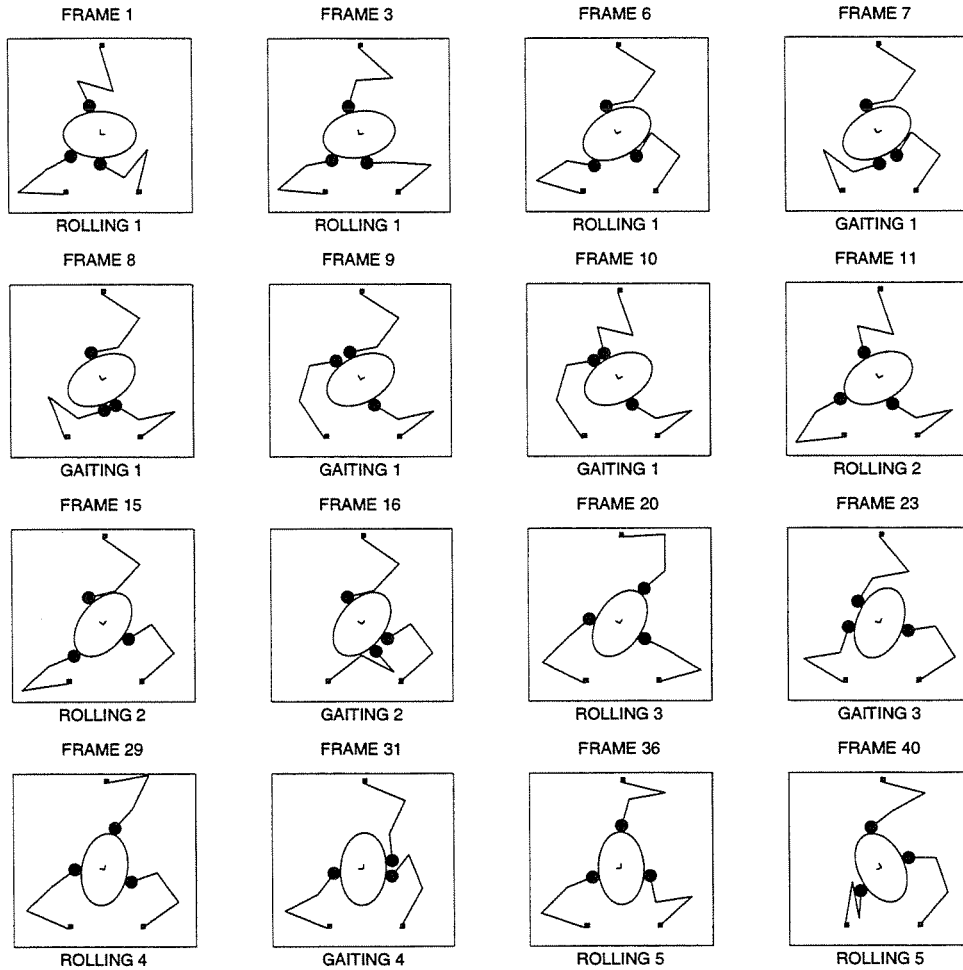


Figure 7.16: Snapshots of a dextrous manipulation motion sequence

7.4.6. Issues Regarding N-Finger Grasps on 3-D Objects

Extending the above method to spatial object grasping is a challenging problem. As shown in Fig. 7.18, 3-D friction cones cannot be expressed as a sum of a finite set of vectors. Therefore, the convex hull condition of a finite set of wrenches cannot be extended in a trivial way to determine if a 3-D grasp is force-closure. Because of this, the proposition saying that the highest FC-contact mode is six also cannot be directly extended to 3-D case. If we assume a point contact model which does not support a torque about the contact normal, the friction cone can be approximated by a polygonal cone. Hence, the method proposed here could be extended in an approximate way to

this case. However, the soft finger contact which supports torque cannot even be handled in this approximate way.

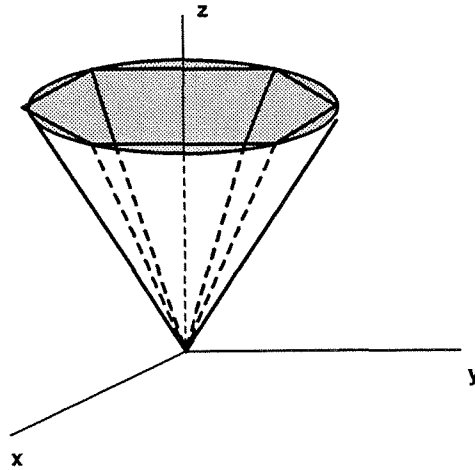


Figure 7.18: Polygonal approximation of a 3-D friction cone

7.5. Discussion

This chapter has presented a generalized approach to planning force-closure grasps on planar and spatial objects in the contact configuration space. The finger contacts are assumed to be point contact with friction (or soft finger contact in spatial object case).

For two-finger grasps, some practical issues in the implementation of antipodal point grasp finding algorithms on 2-D and 3-D smooth nonconvex objects were considered. A simple grasping energy function was introduced, and it has shown that all antipodal points on the object correspond to the critical points of the energy function in the force-closure regions in contact configuration space. Thus, finding the antipodal points is equivalent to a constrained optimization procedure. For the example, a particular global optimization scheme termed TRUST was employed. The analysis included both squeezing and expanding grasps, which occur for non-convex object. This approach can be used with any object whose boundary can be described by continuous functions.

For n -finger grasps, a force-closure test for planar objects was presented. This test is based on the convex hull of the wrenches generated by the point contact friction

cone edge vectors. The critical conditions of this test function were used to define and enumerate the force closure surfaces in an n -dimensional contact C-space. These surfaces enclose regions containing contact configurations which are force-closure. It was also shown that the force-closure sets decompose into regions which correspond to closure of m finger subsets of the n fingers. Certain of these sub-regions, termed gait transition regions, are essential to the implementation of finger gaiting. An algorithm was presented to find these m finger closure subsets. The general methodology is a useful tool for planning roll/slide motions as well. This was demonstrated by a computer simulation. Nevertheless, as mentioned in Section 7.4.6, extending this approach to 3-D objects grasping is still a challenging problem and we hope to consider these problem in future work.

Chapter 8

Conclusions

Fixed configuration robots have already proved their efficiency and accuracy in factory applications. However, as robots are applied to increasingly sophisticated and unstructured tasks, the complexity of robot design grows. The modular reconfigurable robot design provides a feasible solution to increase performance of a robotic system undertaking sophisticated tasks, while limiting the design complexity of the machine through reconfiguration. This thesis addressed a number of modular robot issues that are crucial to modular robotic performance and complexity. The framework developed here provided methodologies to enumerate all possible assembly configurations from an inventory of modules and to determine an optimal one for a task requirement.

The simple set of conceptual module models introduced in this thesis can model existing modular robot implementations. However, the square prism and cube link models are for demonstration purposes. Other symmetric shaped objects, such as triangle prisms and hexagon prisms, can be taken as link modules with their symmetric rotation groups identified. The symmetric link model can be applied to practical link modules without symmetry as well. In this case, the symmetric group of this link contains one element — the identity.

The kinematic graph based representation of modular robots built the foundation for modular robot construction enumeration and synthesis. The basic function of a mod-

ular robot is determined by its graph. The assembly incidence matrix (AIM) fully represents a modular robot assembly configuration, i.e., the port informations and the robot topology. This AIM is best suitable for representing a kinematic chain in which link units have multiple connection methods.

Algebraic and kinematic equivalence relations were defined on the AIMs for classifying physically and kinematically identical modular robot assemblies respectively. Algebraic equivalent AIMs built robot assembly with the same appearance. This equivalence is based on the graph isomorphisms and symmetric rotation groups of the link modules. It is the basis for solving the module assembly enumeration problem. A tree-structured modular robot assembly enumeration algorithm was proposed according to this equivalence relation. This algorithm accepts a prescribed set of link and joint modules. Its output is a set of distinct (non-isomorphic) robot assembly configurations. This algorithm provided modular robot designer a systematic and efficient way to evaluate the assembly capability of an entire set of modules during the initial design phase. Kinematic equivalent AIMs have identical kinematic properties, such as the workspace shape and volume, and joint singularities. This equivalence is based on the equivalence of twist coordinates of the joint axes of the robots. Identifying the kinematic equivalence is crucial for the task-oriented optimal configuration problem when the modular robot kinematics is the main concern. Algebraically equivalent AIMs are kinematically equivalent. The converse does not always hold.

Formulating the task optimal configuration problem as a combinatorial optimization problem presented a feasible way to solve this re-combination problem. Because all the design parameters are pre-determined at the module level, the freedom left in a modular robotic system is through module reconfiguration. A task-related objective function was formed to evaluate the performance of a robot assembly for a task. The domain of the objective function is the set of distinct robot assembly configurations that can be obtained from the assembly configuration enumeration algorithm. This approach can be applied to finding task-related configurations of other mechanical systems with modular design with a properly defined objective function.

Finally, a generalized approach to planning force-closure grasps on planar and spatial objects was presented. The finger contacts are assumed to be point contact with friction (or soft finger contact in spatial object case). For two-finger grasps, we have investigated a set of fast and efficient antipodal point grasps on 2-D and 3-D objects. For n -finger grasps, we introduced a qualitative force-closure test function for planar objects. Applications of this test function to dextrous manipulation tasks were demonstrated. However, extending this test function to spatial objects is still a challenging problem.

Following the context of this thesis, one extension can be made in incorporating closed-loop module constructions in the assembly configuration enumeration algorithm discussed in Chapter 4. An extra procedure to detect a closed-loop topology of the kinematic graph for the algorithm is needed. Mobility analysis of the closed chain and kinematic constraints on closing the loop can be formulated using screw theory.

There are still a number of unexplored topics in the area of task-oriented optimal configurations. A more rigorous rule can be defined for translating kinematic requirements on individual joint assembly patterns into module assembly preferences. Other combinatorial optimization techniques, such as simulated annealing, can be investigated under the same framework. In Chapter 6, the objective function is formulated as a single-valued function. A multi-objective function approach can be explored instead. The objective function thus becomes a vector-valued function which evaluates not only the task performance of the robot but also other goals to be achieved by the robot. A multi-objective genetic algorithm can be employed in this case.

Based on the graph representation, one can explore the autonomous reconfiguration of modular robots, i.e., the determination of the assembling and disassembling sequence of modules from one configuration to the other. Configuration independent dynamical models of the modular robots and control strategy using recursive approach similar to the tree-structured robot forward kinematics derived in Chapter 5 is also necessary for more practical purposes. A more challenging problem is the autonomous determination

of an optimal robot topology for a set of tasks. Robots with distinct topologies function differently. In this thesis, the robot task is assumed to be consistent with the robot topology. To define a robot topology independent task criterion for a task-oriented optimal robot topology problem remains a challenging issue.

The actual design and construction of link modules that are capable of multiple connections is also an important practical issue as well. Most of the currently existing modular robot systems allow only one way to connect joints to a link module. A multi-port design or a universal joint like design is a plausible approach to manifest the multiple connection nature of a link module.

Designing modules with mobility enables a modular robot system to mobile itself or to perform automatic disassembling and reassembling of modules without manual assistance. Many interesting issues arise when mobile modules are introduced in a modular robot system. A modular robotic arm with a mobile base unit has a free choice of the base location. Thus, a robot assembly configuration can perform a variety of tasks just by changing the base location. Coordination between several mobile modules in a modular robot is necessary for self-locomotion. This becomes a robot locomotion and coordination problem.

Our ultimate goal is to provide “intelligence” for every module, i.e., to furnish every module sufficient computation and communication capability such that once these modules are connected together, they can autonomously determine the entire robot assembly configuration and the control scheme for this configuration. Furthermore, they can determine how to reconfigure themselves into one large robot or several smaller robots to perform a task in coordination. This goal will move a modular reconfigurable robotic system toward a distributed intelligent mechanical system. In the long run, we would like to see such a modular reconfigurable robotic system fully integrated into an intelligent work cell in a flexible manufacturing system. In such a system, there is no distinction between CNC machine tools and robot manipulators. Machining tasks are performed by tool modules. Material transporting or welding tasks are done by

automatic reconfiguration of standard modules connected by standardized interfaces. Modules used in different work cells can be exchanged for maintenance and economical reasons. Another place to employ such a system is on a space station. With the standard interface design, one can build a large intelligent reconfigurable structure combining moving parts—the modular robots, and the fixed parts—the truss structure in space. This reconfigurable structure can serve a great many purposes required by a space station, such as repair and maintenance, experimental works in space, or satellite retrieval, in order to reduced the extraterrestrial vehicular activities.

References

- [1] L. Al-Hakim and A. Shrivastava, "Application of Graph Theory for Structural Enumeration and Presentation of Mechanisms," in *Proc. 8th World Congress on the Theory of Machines and Mechanisms*. Prague: pp. 25–28, 1991.
- [2] J. Angeles, "The Design of Isotropic Manipulator Architectures in the Presence of Redundancies," *Int. J. Robotics Research*, 11, no. 3, pp. 196–201, 1992.
- [3] A. H. Barr, "Superquadrics and Angle-Preserving Transformations," *IEEE Computer Graphics and Applications*, 1, pp. 11–23, 1981.
- [4] E. Beckenbach, *Applied Combinatorial Mathematics*. New York, NY, John Wiley & Sons, 1964.
- [5] B. Benhabib, G. Zak and M. G. Lipton, "A Generalized Kinematic Modeling Method for Modular Robots," *J. Robotics Systems*, 6, no. 5, pp. 545–571, 1989.
- [6] D. P. Bertsekas, in *Constrained Optimization and Lagrange Multiplier Methods*. New York, NY: Academic Press, 1982.
- [7] T. Beyer and S. M. Hedetniemi, "Constant Time Generation of Rooted Trees," *SIAM J. Computing*, 9, no. 4, pp. 706–712, 1980.
- [8] H. A. Bremermann, "A method of unconstrained global optimization," *Mathematical Biosciences*, 9, pp. 1–15, 1970.
- [9] R. Brockett, "Robotic Manipulators and the Product of Exponential Formula," in *Proc. MTNS-83 In. Sym.* Beer Sheba, Israel: pp. 120–129, 1983.
- [10] B. Cetin, J. Barhen and J. Burdick, "Terminal Repeller Sub-Energy Tunneling (TRUST) for Fast Global Optimization," *Journal of Optimization Theory and Applications*, 77, no. 1, April, 1993.

- [11] I. -M. Chen and J. W. Burdick, "Finding Antipodal Point Grasps on Irregularly Shaped Objects," in *Proc. IEEE Int. Conf. Robotics and Automation*. Nice, France: pp. 2278–2283, 1992.
- [12] ———, "Finding Antipodal Point Grasps on Irregularly Shaped Objects," *IEEE Trans. Robotics and Automation*, 9, no. 4, pp. 507–512, 1993.
- [13] ———, "A Qualitative Test for N-Finger Force-Closure Grasps on Planar Objects with Application to Manipulation and Finger Gaits," in *Proc. IEEE Int. Conf. Robotics and Automation*. Atlanta, GA: pp. 814–820, 1993.
- [14] ———, "Enumerating Non-Isomorphic Assembly Configurations of Modular Robotic Systems," in *Proc. IEEE/RSJ Int. Workshop Intell. Robots and Systems*. Yokohama, Japan: pp. 1985–1992, 1993.
- [15] G. Chirikjian, "Theory and Applications of Hyper-Redundant Robotic Manipulators," California Institute of Technology, Ph.D. Dissertation, 1992.
- [16] S. L. Chiu, "Task Compatibility of Manipulator Postures," *Int. J. Robotics Research*, 7, no. 5, pp. 13–21, 1988.
- [17] R. Cohen, M. G. Lipton, M. Q. Dai and B. Benhabib, "Conceptual Design of a Modular Robot," *ASME J. Mechanical Design*, 114, pp. 117–125, March, 1992.
- [18] T. Cormen, C. Leiserson and R. Rivest, *Introduction to Algorithms*. Cambridge, MA, MIT Press, 1990.
- [19] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Reading, MA, Addison Wesley, 1986.
- [20] L. Danzer, B. Grünbaum and V. Klee, "Helly's Theorem and Its Relatives," in *Convexity, Proceedings of Symposia in Pure Mathematics*, vol. 7. Providence, RI: American Mathematical Society, pp. 101–180, 1962.

- [21] J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *ASME J. Applied Mechanics*, 2, pp. 215–221, 1955.
- [22] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*. Englewood Cliffs, NJ, Prentice-Hall, 1974.
- [23] E. A. Dinits and M. A. Zaitsev, "Algorithm for the generation of nonisomorphic trees," *Automatic and Remote Control*, 38, pp. 554–558, 1977.
- [24] L. Dobrjanskyj and F. Freudenstein, "Some Applications of Graph Theory to the Structural Analysis of Mechanisms," *ASME J. Engineering for Industry*, 89, pp. 153–158, Feb., 1967.
- [25] C. Earl and J. Rooney, "Some Kinematic Structures for Robot Manipulator Designs," *ASME J. Mech., Trans., and Auto. in Design*, 105, March, 1983.
- [26] S. Even, *Algorithmic Combinatorics*. New York, NY, Macmillan, 1973.
- [27] D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacturing*. Ellis Horwood, 1979.
- [28] B. Faverjon and J. Ponce, "On Computing Two-Finger Force-Closure Grasps of Curved 2D Objects," in *Proc. IEEE Int. Conf. Robotics and Automation*. Sacramento, CA: pp. 424–429, 1991.
- [29] R. Fearing, "Simplified Grasping and Manipulation with Dexterous Robot Hands," *IEEE J. Robotics and Automation*, 2, no. 4, 1986.
- [30] J. B. Fraleigh, *A First Course in Abstract Algebra* 3ed. Reading, MA, Addison Wesley, 1982.

- [31] F. Freudenstein, "The Basic Concepts of Pólya Theory of Enumeration, with Application to the Structural Classification of Mechanisms," *J. Mechanisms*, 3, pp. 275–290, 1967.
- [32] T. Fukuda and S. Nakagawa, "Dynamically Reconfigurable Robotic System," in *Proc. IEEE Int. Conf. Robotics and Automation*. pp. 1581–1586, 1988.
- [33] T. Fukuda, S. Nakagawa, Y. Kawauchi and M. Buss, "Structure Decision Method for Self Organizing Robots Based on Cell Structures-CEBOT," in *Proc. IEEE Int. Conf. Robotics and Automation*. pp. 695–700, 1989.
- [34] T. Fukuda, T. Ueyama and F. Arai, "Control Strategy for a Network of Cellular Robots," in *Proc. IEEE Int. Conf. Robotics and Automation*. pp. 1616–1621, 1991.
- [35] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA, Addison Wesley, 1989.
- [36] K. C. Gupta, "On the Nature of Robot Workspace," *Int. J. Robotics Research*, 5, no. 2, pp. 112–121, 1986.
- [37] F. Harary, *Graph Theory*. Reading, MA, Addison-Wesley, 1972.
- [38] F. Harary and H. Yan, "Logical Foundations of Kinematic Chains: Graphs, Line Graphs, and Hypergraphs," *ASME J. Mechanical Design*, 112, pp. 79–83, March, 1990.
- [39] I. N. Herstein, *Topics in Algebra* 2ed. New York, NY, John Wiley & Sons, 1975.
- [40] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

- [41] J. Hong, G. Lafferriere, B. Mishra and X. Tan, "Fine Manipulation with Multifinger Hands," in *Proc. IEEE Int. Conf. Robotics and Automation*. Cincinnati, OH: pp. 1568–1573, 1990.
- [42] P. Hsu, Z. Li and S. Sastry, "On Grasping and Coordinated Manipulation by a Multifingered Robot Hand," *Int. J. Robotics Research*, 8, no. 4, 1989.
- [43] K. H. Hunt, *Kinematic Geometry of Mechanisms*. New York, NY, Oxford Univ. Press, 1978.
- [44] R. Johnsonbaugh, *Discrete Mathematics*. New York, NY, Macmillan, 1984.
- [45] H. B. Keller, in *Lectures on Numerical Methods in Bifurcation Problems*. New York, NY: Springer-Verlag, 1987.
- [46] L. Kelmar and P. Khosla, "Automatic Generation of Kinematics for a Reconfigurable Modular Manipulator System," in *Proc. Int. Conf. Robotics and Automation*. pp. 663–668, 1988.
- [47] P. K. Khosla, C. P. Neuman and F. B. Prinz, "An Algorithm for Seam Tracking Applications," *Int. J. Robotics Research*, 4, no. 1, pp. 27–41, 1985.
- [48] J. O. Kim and P. Khosla, "Design of Space Shuttle Tile Servicing Robot: An Application of Task Based Kinematic Design," in *Proc. IEEE Int. Conf. on Robotics and Automation*. pp. 867–874, 1993.
- [49] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, 220, pp. 671–680, 1983.
- [50] C. A. Klein and B. E. Blaho, "Dexterity Measures for the Design and Control of Kinematically Redundant manipulators," *Int. J. Robotics Research*, 6, no. 2, pp. 72–83, 1987.

- [51] C. A. Klein and T. A. Miklos, "Spatial Robotic Isotropy," *Int. J. Robotics Research*, 10, no. 4, pp. 426–437, 1991.
- [52] A. V. Kozina, "Coding and generation of nonisomorphic trees," *Cybernetics*, 15, pp. 645–651, 1975.
- [53] N. H. Kuiper, "Double Normals of a Convex Body," *Israel J. of Mathematics*, 2, pp. 71–80, 1964.
- [54] J. -C. Latombe, *Robot Motion Planning*. Boston, MA, Kluwer Academic Publishers, 1991.
- [55] T. W. Lee and D. C. H. Yang, "On the Evaluation of Manipulator Workspace," *ASME J. Mech. Trans. Automation*, 105, pp. 70–77, March, 1983.
- [56] Z. Li, "Geometrical Considerations of Robot Kinematics," *Int. J. Robotics and Automation*, 5, no. 3, pp. 139–145, 1990.
- [57] Z. Li and J. Canny, "Motion of Two Rigid Bodies with Rolling Constraint," *IEEE Trans. Robotics and Automation*, 6, no. 1, 1990.
- [58] Z. Li, J. Canny and S. Sastry, "On Motion Planning for Dexterous Manipulation, Part I: The Problem Formulation," in *Proc. IEEE Int. Conf. on Robotics and Automation*. pp. 775–780, 1989.
- [59] C. L. Liu, *Introduction to Combinatorial Mathematics*. New York, NY, McGraw-Hill, 1968.
- [60] X. Markenscoff and C. H. Papadimitriou, "Optimum grip of a polygon," *Int. J. Robotics Research*, 8, no. 2, pp. 17–29, 1989.
- [61] M. Mason and J. K. Salisbury, *Robot Hands and the Mechanics of Manipulation*. Cambridge, MA, MIT Press, 1985.

- [62] J. M. McCarthy, *Introduction to Theoretical Kinematics*. Cambridge, MA, MIT Press, 1990.
- [63] C. Mead, *Introduction to VLSI systems*. Reading, MA, Addison Wesley, 1980.
- [64] W. Miller, *Symmetry Groups and Their Applications*. New York, NY, Academic Press, 1972.
- [65] B. Mishra, J. Schwartz and M. Sharir, "On the Existence and Synthesis of Multifinger Positive Grips," *Algorithmica*, 2, pp. 541–558, 1987.
- [66] D. J. Montana, "The Kinematics of Contact and Grasp," *Int. J. Robotics Research*, 7, no. 3, pp. 17–32, 1988.
- [67] ———, "Contact Stability for Two-Fingered Grasps," *IEEE Trans. Robotics and Automation*, 8, no. 4, pp. 421–430, 1992.
- [68] R. E. Moore, in *Interval Analysis*. Englewood Cliffs, NJ: Prentice Hall, 1966.
- [69] R. Murray, Z. Li and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.
- [70] R. Murray and S. Sastry, "Grasping and Manipulation Using Multifingered Robot Hands," in *Robotics, Proc. Symposia in Applied Mathematics*, vol. 41. Providence, RI: American Mathematical Society, pp. 91–127, 1990.
- [71] Y. Nakamura and H. Hanafusa, "Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control," *J. Dynamic System, Measurement, and Control*, 108, pp. 163–171, Sep., 1986.
- [72] V. Nguyen, "Constructing Force-Closure Grasps," *Int. J. Robotic Research*, 7, no. 3, pp. 3–16, 1988.

- [73] A. Nijenhuis and H. Wilf, *Combinatorial Algorithms*. New York, NY, Academic Press, 1975.
- [74] T. Omata, "Fingertip Positions of a Multifingered Hand," in *Proc. IEEE Int. Conf. on Robotics and Automation*. Cincinnati, OH: pp. 1562–1567, 1990.
- [75] B. Paden and S. Sastry, "Optimal Kinematic Design of 6R Manipulators," *Int. J. Robotics Research*, 7, no. 2, pp. 43–61, 1988.
- [76] C. Paredis and P. Khosla, "Kinematic Design of Serial Link Manipulators From Task Specifications," *Int. J. Robotics Research*, 12, no. 3, pp. 274–287, 1993.
- [77] C. J. Paredis and P. K. Khosla, "Mapping Tasks into Fault Tolerant Manipulators," presented at IEEE Int. Conf. Robotics and Automation, San Diego, CA, 1994.
- [78] F. Park, "On the Optimal Kinematic Design of Spherical and Spatial Mechanisms," in *Proc. IEEE Int. Conf. Robotics Automation*. Sacramento, CA: pp. 1530–1535, 1991.
- [79] ———, "The Optimal Kinematic Design of Mechanisms," Harvard Univ., Ph.D. Dissertation, 1991.
- [80] F. Park, A. Murray and J. McCarthy, *Designing Mechanisms for Workspace Fit*, 1993, preprint.
- [81] Y. C. Park and G. P. Starr, "Grasp Synthesis of Polygonal Objects," in *Proc. IEEE Int. Conf. Robotics and Automation*. Cincinnati, OH: pp. 1574–1580, 1990.
- [82] A. Pentland, "Perceptual Organization and the Representation of Natural Form," *Artificial Intelligence*, 28, pp. 293–331, 1986.
- [83] A. Pentland and R. Bolles, "Learning and Recognition in Natural Environments," in *Robotics Science*, M. Brady, Ed. Cambridge, MA: MIT Press, pp. 164–207, 1989.

- [84] M. Raghavan, "Manipulator Kinematics," in *Robotics, Proc. Symposia in Applied Mathematics*, vol. 41. Providence, RI: American Mathematical Society, 1990.
- [85] M. Raghavan and B. Roth, "Inverse Kinematics of the General 6R Manipulator and Related Linkages," *ASME J. Mechanical Design*, 115, pp. 502–508, Sep., 1993.
- [86] R. C. Read, "How to grow trees," in *Combinatorial Structures and Their Applications*. New York, NY: Gordon and Breach, 1970.
- [87] D. Schmitz, P. Khosla and T. Kanade, "The CMU Reconfigurable Modular Manipulator System," Carnegie Mellon Univ., CMU-RI-TR-88-7, 1988.
- [88] P. Sheth and J. Uicker, "A Generalized Symbolic Notation for Mechanisms," *ASME J. Engineering for Industry*, 93, no. 1, pp. 102–112, Feb., 1971.
- [89] S. Skiena, *Implementing Discrete Mathematics*. Redwood City, CA, Addison-Wesley, 1990.
- [90] D. Stewart, R. Volpe and P. Khosla, "Integration of Real-Time Software Control Modules for Reconfigurable Sensor-based Systems," in *Proc. IEEE/RSJ Int. Workshop Intell. Robots and Systems*. 1992.
- [91] D. Tesar and M. S. Butler, "A Generalized Modular Architecture for Robot Structures," *ASME J. of Manufacturing Review*, 2, no. 2, pp. 91–117, 1989.
- [92] L. -W. Tsai and A. P. Morgan, "Solving the Kinematics of the Most General 6 and 5 DOF Manipulators by Continuation Methods," *ASME J. Mech. Trans. Automation*, 107, pp. 189–200, June, 1985.
- [93] Y. C. Tsai and A. H. Soni, "The Effect of Link Parameter on the Working Space of General 3R Robot Arms," *Mechanism and Machine Theory*, 19, no. 1, pp. 9–16, 1984.

- [94] T. Ueyama, T. Fukuda and F. Arai, "Configuration of Communication Structure for Distributed Intelligent Robotic System," in *Proc. IEEE Int. Conf. Robotics and Automation*. pp. 807–812, 1992.
- [95] ———, "Structure Configuration Using Genetic Algorithm for Cellular Robotic System," in *Proc. of IEEE/RSJ Int. Workshop Intell. Robots and Systems*. Raleigh, NC: pp. 1542–1549, 1992.
- [96] R. Vijaykumar, K. Waldron and M. Tsai, "Geometric Optimization of Serial Chain Manipulator Structures for Working Volume and Dexterity," *Int. J. Robotics Research*, 5, no. 2, pp. 91–103, 1986.
- [97] C. W. Wampler, A. P. Morgan and A. J. Sommese, "Numerical Continuation Methods for Solving Polynomial Systems Arising in Kinematics," *ASME J. Design*, 112, pp. 59–68, March, 1990.
- [98] A. B. Wells, "Grammars for Engineering Design," California Institute of Technology, Ph.D. Dissertation, 1994.
- [99] D. E. White and S. G. Williamson, "Construction of Minimal Representative Systems," *Linear and Multilinear Algebra*, 9, pp. 167–180, 1980.
- [100] S. Williamson, *Combinatorics for Computer Science*. Rockville, MD, Computer Science Press, 1985.
- [101] L. S. Woo, "Type Synthesis of Plane Linkages," *ASME J. Engineering for Industry*, 89, pp. 159–172, Feb., 1967.
- [102] R. A. Wright, B. Richmond, A. Odlyzko and B. D. McKay, "Constant Time Generation of Free Trees," *SIAM J. Computing*, 15, no. 2, pp. 540–548, 1986.
- [103] K. H. Wurst, "The Conception and Construction of a Modular Robot System," in *Proc. 16th Int. Sym. Industrial Robotics (ISIR)*. pp. 37–44, 1986.

- [104] F. Yamaguchi, *Curves and Surfaces in Computer Aided Geometry Design*. New York, NY, Springer-Verlag, 1988.
- [105] H. -S. Yan and Y. -W. Hwang, "The specialization of Mechanisms," *Mechanism and Machine Theory*, 26, no. 6, pp. 541-551, 1991.
- [106] D. C. H. Yang and T. W. Lee, "On the Workspace of Mechanical Manipulators," *ASME J. Mech. Trans. Automation*, 105, pp. 62-69, March, 1983.
- [107] T. Yoshikawa, "Analysis and Control of Robot Manipulators with Redundancy," in *Robotics Research: The First Int. Sym.*, M. Brady and R. Paul, Eds. MIT Press, pp. 735-747, 1984.
- [108] ———, "Manipulability of Robotic Mechanisms," *Int. J. Robotics Research*, 4, no. 2, pp. 3-9, 1985.