Kinematics, Dynamics, Calibration, and Configuration Optimization of Modular Reconfigurable Robots

Yang Guilin

Submitted in Fulfillment of
the Requirements for the Degrees of
Doctor of Philosophy

School of Mechanical and Production Engineering
Nanyang Technological University

January 1999

Abstract

A modular reconfigurable robot system (MRRS) is a collection of individual link and joint units that can be assembled into various robot configurations for a diversity of task requirements. Such a modularized robotic system can be applied in various scenarios: an individual joint unit can be used as an independent motion control element; fully assembled systems can be used as standard robots with various configurations and *degrees of freedom* (DOFs). This thesis focuses on the issues pertinent to an MRRS: design, modeling, and optimization.

Module Design and Modular Robot Representation: An inventory of conceptual robot modules - joint and link modules, has been designed. With these modules, many possible robot configurations can be constructed. A kinematic graph based representation technique, termed Assembly Incidence Matrix (AIM), is then employed to mathematically represent various robot configurations for automatic model generations.

Automatic Model Generations: The MRRS has unfixed configurations so that developing a set of configuration-independent kinematics, dynamics, and calibration algorithms for configuration optimization, motion planning, and control becomes the main concern of this thesis. For this purpose, a geometric modeling method based on the local presentation of *product-of-exponentials* (POE) formula is employed. The proposed algorithms can generate the kinematic, dynamic, and calibration models for a general tree-structured robot configuration automatically, regardless of the types of joints and the number of DOFs. The effectiveness of these algorithms is demonstrated by simulation examples.

Configuration Optimization: The MRRS can provide many workable robot configurations for a specific task. With fewer numbers of modules and DOFs, a modular robot would have a simple configuration and a low power consumption rate, so that it can perform the designated task effectively. The *minimized degrees of freedom* (MDOF) concept is thus introduced in optimizing the robot configurations. Based on the effective coding schemes and the virtual module concept, an evolutionary algorithm (EA) approach is employed to search for the optimal solutions. This design methodology is demonstrated by computation examples.

A unified geometric framework that addresses most of the aspects of modular robot problem has been synthesized in this thesis. By taking tools from graph theory and differential geometry, this geometric framework provides an integrated and user-friendly environment for analysis and design of modular robots.

Acknowledgment

I would like to express my sincere gratitude to my advisor, Dr Chen I-Ming, for his continuous interest, intellectual guidance, infinite patience, and constant encouragement during this research. His vision and broad knowledge play an important role in the realization of this work.

I would like to thank Professor Ling Shih Fu, Dr Yeo Song Huat, and Dr Zribi Mohamod, for their valuable comments on my confirmation examination. Special thanks are given to Professor Park Chongwoo of Seoul National University, Korea, and Professor Joel Burdick of California Institute of Technology, USA, for their insightful advice and encouragement. I am also grateful to Dr Du Hejun, Dr Chen Chaoyu, Dr Chen Guang, Dr Low Kin Huat, and Dr Xiao Pengdong for their support and help.

I would also like to thank Dr Gerald Seet, Mr Lim Eng Cheng, Ms Agnes Tan, and Mr Yow Kim San for providing an excellent research environment at the Robotics Research Centre. I also want to thank my colleague and friend Mr Tan Chee Tat for his computation assistance and the endless discussions. Thanks are also extended to my long time fellow graduate students, Liu Bao, Xiang Zhonggui, Xu Tongge, Huang Tao, and Chen Wenjie for their friendship, encouragement, and help.

I would like to express my gratitude to all my family members, especially my parents, my wife, and my daughter for their unconditional love and support.

Finally, I would also like to acknowledge the financial support that I received during my study here. My scholarship has been sponsored by Nanyang Technological University, and the research work is partially supported by Gintic Institute of Manufacturing Technology, Singapore, under the upstream project U-97-A006.

List of Symbols

Ad_X	Adjoint presentation associated with X, $(X \in SE(3))$
ad_x	Adjoint presentation associated with $x(x \in se(3))$
$\mathcal{B}(\vec{G})$	Adjacency matrix of a directed graph $\vec{G}$
$d\hat{p}_i$	Kinematic error in $T_{i-1,i}(0)$ expressed in module frame i, $d\hat{p}_i \in se(3)$
$d\hat{p}_{i-1,i}$	Kinematic error in $T_{i-1,i}(0)$ expressed in module frame $i-1$, $d\hat{p}_{i-1,i} \in se(3)$
$e^{\hat{s}}$	Exponential of $\hat{s} \in se(3)$, $e^{\hat{s}} \in SE(3)$
F_i	Resultant wrench applied to link assembly i, $F_i = (f_i, \tau_i)^T \in \Re^{6\times 1}$
f_i	Resultant force applied to link assembly i, $f_i \in \Re^{3\times 1}$
J	Generalized body manipulator Jacobian
J_{fi}	Fixed joint of size $i(i = 1, 2, 3)$
J_{pi}	Prismatic joint of size $i(i = 1, 2, 3)$
J_{ri}	Revolute joint of size $i(i = 1, 2, 3)$
J_v	Virtual joint
$\log(T)$	Logarithm of $T \in SE(3)$, $\log(T) \in se(3)$
$\log(T)^\vee$	Vector representation of $\log(T)$, $\log(T)^\vee \in \Re^{6\times 1}$
L_{ci}	Cubic link of size $i(i = 1, 2)$
L_{pi}	Prismatic link (derived from prismatic joint module) of size $i(i = 1, 2)$
L_{ri}	Revolute link (derived from revolute joint module) of size $i(i = 1, 2)$
L_v	Virtual link
H	Transmission matrix
$\mathcal{M}(\vec{G})$	Incidence matrix of a directed graph $\vec{G}$
M_i	Generalized mass matrix of link assembly i, $M_i \in \Re^{6\times 6}$
m_i	Mass of link assembly i
p	Position vector, $p \in \Re^{3\times 1}$
$\mathcal{P}(\vec{G})$	Path matrix of a directed tree $\vec{G}$
q_i	Joint i's displacement
$\dot{q}_i$	Joint i's velocity
$\ddot{q}_i$	Joint i's acceleration
R	Rotation Matrix, $R \in SO(3)$
$\mathcal{R}(\vec{G})$	Accessibility matrix of a directed graph $\vec{G}$
$\hat{s}_i$	Twist of joint i expressed in module frame i, $\hat{s}_i \in se(3)$
$\hat{s}_{i,j}$	Twist of joint j expressed in module frame i, $\hat{s}_{i,j} \in se(3)$
τ_i	Resultant moment (torque) applied to link assembly i, $\tau_i \in \Re^{3\times 1}$
T	4×4 homogeneous kinematic transformation matrix, $T \in SE(3)$
T^{-1}	Inverse matrix of T, $T^{-1} \in SE(3)$
T^T	Transpose matrix of T, $T^T \in SE(3)$
$T_{i,j}$	Kinematic transformation from frame i to frame j, $T_{i,j} \in SE(3)$
$T_{i,j}(0)$	Initial pose of frame j with respect to frame i, $T_{i,j}(0) \in SE(3)$
$T^c_{i-1,i}(0)$	Calibrated initial pose of module frame i with respect to module frame $i-1$, $T_{i-1,i}(0) \in SE(3)$
$T^a_{0,n+1}$	Actual (measured) pose of the tool frame with respect to the base frame, $T^a_{0,n+1} \in SE(3)$
V_i	Generalized body velocity of link assembly i, $V_i \in \Re^{6\times 1}$
$\dot{V}_i$	Generalized body acceleration of link assembly i, $\dot{V}_i \in \Re^{6\times 1}$

List of Abbreviation

AIM	Assembly Incidence Matrix
BFS	Breadth-first Search
CI	Condition Index
CM	Center of the Mass
DFS	Depth-first Search
D-H	Denavit-Hartenberg
DOFs	Degrees of Freedom
EA	Evolutionary Algorithm
eIM	Extended Incidence Matrix
GA	Generic Algorithm
L-E	Lagrange-Euler
MDOF	Minimized Degrees of Freedom
MRRS	Modular Reconfigurable Robot System
N-E	Newton-Euler
POE	Product-of-Exponentials
SA	Simulated Annealing Algorithm
$SE(3)$	Special Euclidean Group
$se(3)$	Lie Algebra of $SE(3)$
$SO(3)$	Special Orthogonal Group
$so(3)$	Lie Algebra of $SO(3)$
TBD	Task Based Design

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Past Research Efforts 3

1.3 Overview of this Thesis 10

1.3.1 Main Contributions 10

1.3.2 Organization 13

2 Conceptual Module Design **14**

2.1 Module Design Requirements 14

2.2 Joint Modules 15

2.2.1 Revolute Joint Modules 15

2.2.2 Prismatic Joint Modules 17

2.3 Link Modules 19

2.4 Connectors 21

2.5 Discussion 23

3 Modular Robot Representation **24**

3.1 Graphs 25

3.1.1 Basic Graph Definitions 25

3.1.2 Matrix Representation of Graphs 27

3.2 Kinematic Graphs 33

3.3 Re-classification of Links and Joints 34
3.4 Assembly Incidence Matrix 35
3.5 Discussion 38

4 Modular Robot Kinematics 39

4.1 Introduction 40
4.2 Geometric Background and the POE Formula 44
4.2.1 Geometric Background 44
4.2.2 The POE Formula 46
4.3 Forward Kinematics 48
4.3.1 Dyad Kinematics 48
4.3.2 Forward Kinematics for a Tree-structured Modular Robot 53
4.4 Inverse Kinematics 55
4.4.1 Differential Kinematics Model for a Single Branch 56
4.4.2 Differential Kinematics Model for a Tree-structured Robot 59
4.4.3 Computation Examples 63
4.4.4 Remarks on Computation Results 65
4.5 Discussion 67

5 Modular Robot Dynamics 68

5.1 Introduction 69
5.2 Newton-Euler Equation for a Link Assembly 71
5.3 Dynamic Formulation for a Tree-structured Modular Robot 74
5.3.1 Recursive Newton-Euler Algorithm 74
5.3.2 Closed Form Equations of Motion 77
5.3.3 Remarks on the Dynamics Algorithms 78
5.3.4 Implementation and Examples 79

5.4 Inverse and Forward Dynamics Problem 82
5.4.1 Inverse Dynamics 82
5.4.2 Forward Dynamics 83
5.5 Discussion 87

6 Kinematic Calibration for Modular Robots 88
6.1 Introduction 89
6.2 Kinematic Calibration Models 92
6.2.1 Basic Calibration Models 94
6.2.2 An Iterative Least-Squares Algorithm 97
6.2.3 Kinematic Calibration of Tree-structured Robots 99
6.3 Computation Examples 102
6.3.1 Calibration of a 3-module Robot 102
6.3.2 Calibration of a SCARA Type Robot 106
6.3.3 Calibration of a Tree-structured Robot 108
6.4 Discussion 111

7 Optimization of Modular Robot Configurations 112
7.1 Introduction 113
7.2 General Design Methodology 118
7.3 Optimization Model 120
7.3.1 Definition of Robot Tasks 120
7.3.2 Design Parameters and the Search Space 121
7.3.3 The Objective Function 124
7.3.4 Performance Constraints 125
7.4 Evolutionary Algorithm 128
7.4.1 Coding Scheme 128

7.4.2 AIM Generating Scheme 129

7.4.3 Genetic Operators on AIMs 129

7.4.4 Implementation of the Evolutionary Algorithm 131

7.5 Computation Examples 133

7.6 Discussion 144

8 Conclusions 145

8.1 Contributions 145

8.2 Future Directions 148

A Generalized Mass Matrix for a Link Assembly 152

A.1 Coordinate System in a Link Assembly 152

A.2 Kinematic Transformation from Frame j to i 153

A.3 Determination of the CM of a Link Assembly 153

A.4 Generalized Mass Matrix of a Link Assembly 154

B Geometric Interpretation of the Kinematic Errors 156

List of Figures

1.1 Application scenarios of different robotics and automation systems 2

1.2 An RMMS link and pivot joint module (adapted from [93]) 4

1.3 A robot configuration constructed by RMMS modules (adapted from [93]) 4

1.4 Robot modules in the MRS (adapted from [29]) 5

1.5 A SCARA type modular robot configuration constructed by MRS modules (adapted from [29]) 6

1.6 Three different types of rotations 7

1.7 Three different (2-DOF) RMD-1 assembly configurations 8

1.8 MoRSE drive modules 8

1.9 A PUMA type robot configuration constructed by MoRSE modules 9

1.10 Relationships among the research issues 11

2.1 Revolute joint module 16

2.2 FH-2000 series motor 17

2.3 Coordinate frames in a revolute joint module 18

2.4 Prismatic joint module 18

2.5 Coordinate frames in a prismatic joint module 19

2.6 Cubic link module 20

2.7 Coordinate frames in cubic link module 21

2.8 Connector/adapter 21

2.9 Coordinate system in a connector/adapter 22

2.10 Electronic connector 23

3.1 A graph and a directed graph 25
3.2 A connected graph and a tree 26
3.3 A labeled graph and a specialized graph 27
3.4 A tree and a directed tree 31
3.5 A planar 6-bar linkage and its kinematic graph 33
3.6 Redefined links 35
3.7 A modular robot assembly configuration 36
3.8 Relative orientations of a joint attaching to a link's connecting port 37

4.1 Two connected modules (a dyad) 49
4.2 Determination of the initial pose of a dyad 51
4.3 A tree-structured modular robot configuration (9-DOF) 54
4.4 Flow chart of iterative inverse kinematics algorithm 60
4.5 A 6-DOF serial robot configuration 65

5.1 Coordinate system in a link assembly 72
5.2 Flow chart of dynamic model generation 80
5.3 A tree-structured modular robot configuration (5-DOF) 80
5.4 Flow chart of the forward dynamic algorithm 83
5.5 Joint acceleration 84
5.6 Joint velocity 85
5.7 Joint displacement 86
5.8 Robot postures at different time instants 86

6.1 Coordinate frames in a dyad 93
6.2 Coordinate frames in the end-effector 93

6.3 Iterative calibration algorithm 99
6.4 A tree-structured modular robot configuration (7-DOF) 100
6.5 A 3-module robot configuration (2-DOF) 102
6.6 Calibration convergence (3-module robot) 105
6.7 A SCARA type modular robot configuration 106
6.8 Calibration convergence (SCARA robot) 107
6.9 Calibration convergence for branch one 109
6.10 Calibration convergence for branch two 110

7.1 Flow chart of the iterative design methodology 119
7.2 Virtual module in the kinematic structure 122
7.3 Crossover operation 130
7.4 Mutation operation 130
7.5 The kinematic design algorithm 131
7.6 Task evaluation procedure 132
7.7 Robot configurations in the initial generation (position problem) 134
7.8 Robot configurations in the final generation (position problem) 135
7.9 Average fitness value in every generation (position problem) 135
7.10 Minimal fitness value in every generation (position problem) 135
7.11 Optimal robot configuration (position problem) 136
7.12 Execution of the given task with the optimal robot configuration (position problem) 136
7.13 Sub-optimal robot configurations (position problem) 136
7.14 Robot configurations in the initial generation (pose problem) 139
7.15 Robot configurations in the final generation (pose problem) 140
7.16 Average fitness value in every generation (pose problem) 140
7.17 Minimal fitness value in every generation (pose problem) 141

7.18 Optimal robot configuration (pose problem) 141

7.19 Execution of the given task with the optimal robot configuration (pose problem) 141

7.20 Sub-optimal robot configurations (pose problem) 142

A.1 Coordinate system in a link assembly 153

B.1 Kinematic errors in a dyad 156

List of Tables

2.1 Technical specifications of revolute joint modules 17

2.2 Technical specifications of prismatic joint modules 19

2.3 Technical specifications of cubic link modules 20

2.4 Technical specifications of connectors/adapter 22

2.5 Technical specifications of connectors/adapter (including the moving part of the prismatic joint module) 23

4.1 Computation results 66

5.1 Computational complexity 79

5.2 Joint acceleration ($1/s^2$ or m/s^2) 85

5.3 Joint velocity ($1/s$ or m/s) 85

5.4 Joint displacement (rad or m) 85

6.1 Preset geometric errors for the 3-module robot 104

6.2 Identified kinematic errors for the 3-module robot 104

6.3 End link poses before and after calibration (3-module robot) 104

6.4 Preset geometric errors for the SCARA robot 107

6.5 Identified kinematic errors for the SCARA robot 107

6.6 End link poses before and after calibration (SCARA robot) 108

6.7 Preset geometric errors for the tree-structured robot 109

6.8 Identified kinematic errors for the tree-structured robot 109

6.9 End link poses before and after calibration (tree-structured robot) 110

7.1 Task points 133

7.2 The adaptive probabilities for the mutation operators (position problem) . 134

7.3 Task poses 138

7.4 The adaptive probabilities for the mutation operators (pose problem) . . . 139

Chapter 1

Introduction

1.1 Motivation

Modularity is a design concept that enables the designer to design separate functional units (or modules) and to use these units individually or collectively, so as to form an integrated working system. Such a design concept has been seen in many engineering disciplines: mechanical engineering, electrical and electronics engineering, and computer and software engineering. Modularity will bring in the capability of flexibility, rapid change-over, and ease-of-maintenance. A system designed in such a way would be reconfigurable, scaleable, and reusable. From the system level aspect, the overall performance may not be optimized although the individual unit design is optimal. More integration effort, for instance, the compatibility between different units, has to be made on modularly designed systems.

Traditionally, robotics and automation systems are used extensively in the structured environment such as the factory floor for various manufacturing purposes. Nowadays, they also make their ways into other areas, such as medical and health care applications, hazardous environment exploration and material handling, security, military, law enforcement, and entertainment. In the design and selection of an effective robotics and automation system for a set of tasks (services), however, system flexibility and productive capacity are always the main concern. Fig. 1.1 shows the application scenarios of different robotics and automation systems based on the variety and volume (or quantity) of the tasks (services). For high-volume/low-variety tasks (services), fully customized equipment is preferred because the capital investment can be justified by the reliability

and productivity provided by the system. For low-volume/high-variety tasks (services), fully general-purpose machines can be employed because the high profit margin can cover the capital investment and maintenance cost. However, there is a large area of tasks (services) that belongs to the mid-volume/mid-variety range, which usually is a compromise between flexibility and productivity. In this case, employing modularly designed robotics and automation systems becomes an attractive solution. This is because (1) the flexibility of the system can be enabled through the reconfiguration of the modular units, and at the same time (2) the high level of productivity can be achieved through the optimally designed modular workcells that provide high operational efficiency.

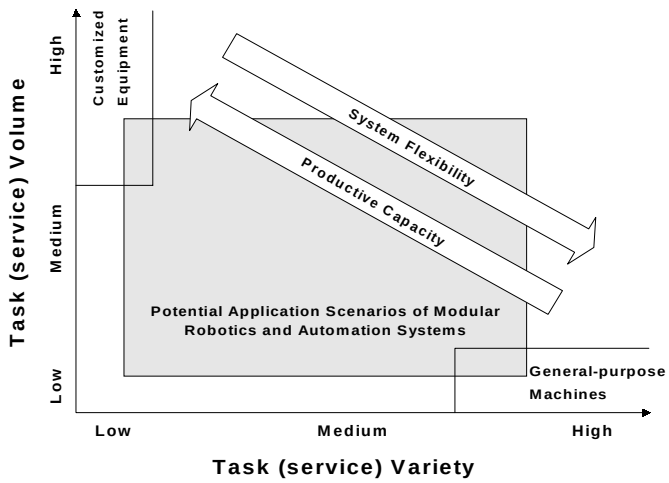


Figure 1.1: Application scenarios of different robotics and automation systems

This thesis mainly discusses issues related to modular reconfigurable robot systems. A modular reconfigurable robot system (MRRS) is a collection of individual link and joint units that can be assembled into various robot configurations for a diversity of task requirements. Possessing the advantage mentioned above, such a modularized robotic system has a variety of application scenarios: an individual actuator unit can be used as a self-contained motion control element; fully assembled systems can be used as standard robots with a variety of configurations and DOFs. Here we focus on the fully assembled systems – the modular robots. Many research issues, such as module design, kinematic and dynamic modeling, calibration, and configuration optimization, arise during the integration of the modules into deployable and effective robot systems.

1.2 Past Research Efforts

There has been an increasing interest in designing modularized robot systems in recent years. The research efforts include system hardware design [1, 2, 7, 17, 29, 42, 51, 84, 94, 115, 124, 132], kinematics [8, 14, 17, 25, 22, 66, 77], dynamics [18, 19, 20, 25], control [84, 93, 115], calibration [25, 21, 122, 136, 137], and task-based design of modular robot configurations [14, 15, 16, 28, 43, 51, 69, 93, 95, 96, 138]. This section briefly reviews the fundamental research efforts on modular robotics. Literature surveys on the individual topics will be presented in the corresponding chapters.

The most fundamental issue in modular robotics is the system hardware design including both mechanical and electronic designs. Up to date, several prototype systems have been demonstrated in various research institutions, such as the Reconfigurable Modular Manipulator System (RMMS) at Carnegie Mellon University [94, 115], the Dynamically Reconfigurable Robotic System (DRRS) at the Science University of Tokyo [42], the TOshiba Modular Manipulator System (TOMMS) at Toshiba Corp. [84], and other modular robot systems at University of Toronto [29], University of Stuttgart [132], and University of Texas [1, 124].

The RMMS at Carnegie Mellon University [115, 94, 93] consists two types of 1-DOF revolute joints: "rotate" and "pivot" joints, and links of different sizes (Fig 1.2). The revolute joints are actuated by DC motors in conjunction with harmonic-drive mechanisms. The links have cylindrical shapes. The mechanical coupling uses the V-band clamp, which is an integral part of link and joint modules. In the second generation of RMMS [94, 93], a quick-coupling mechanism is designed for easy module connection. A RMMS robot configuration is shown in Fig. 1.3. The modules are communicated with a VME-based host controller over a local area network called the ARMbus. The control software for the RMMS has been developed using the Chimera real-time operating system. Because closed-form inverse kinematics solutions do not exist for all possible RMMS configurations, a damped least-squares kinematic control algorithm is employed. A number of research issues are addressed extensively for the RMMS: the automatic generation of kinematics [66], the global fault tolerant planning [97, 93], and the task-based design problem

[69, 95, 96, 93, 98].

(a) link (b) Pivot Joint

Figure 1.2: An RMMS link and pivot joint module (adapted from [93])

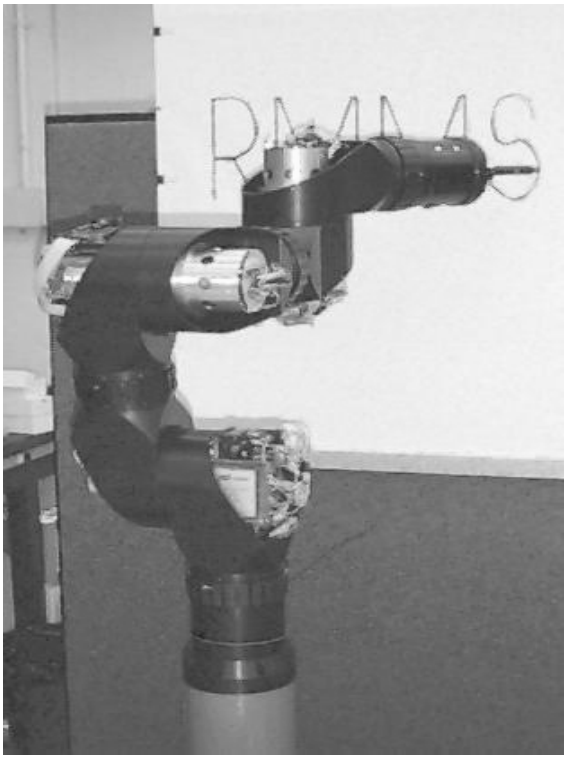


Figure 1.3: A robot configuration constructed by RMMS modules (adapted from [93])

The modular robotic system (MRS) at University of Toronto [29] includes two types of 1-DOF main joint modules (rotary and prismatic), three types of 1-DOF end-effector joint modules (yaw, pitch, and roll type), and link modules with hollow cylindrical geometry (Fig. 1.4). All the joint modules are designed to be self-contained independent units.

A revolute joint module driven by a DC motor with a harmonic-drive mechanism can generate rotary motion; a prismatic joint module driven by a DC motor with a ball-screw mechanism can generate linear motion. A 4-DOF SCARA type modular robot configuration constructed by using these modules is shown in Fig. 1.5. A unique feature of this system is the development of a 45-degree (relative to the modules' main axes) connection structure. This structure allows both types of straight and perpendicular connections by simply rotating the next module by 180 degrees about its main axis. Based on the Denavit-Hartenberg parameter notation, a configuration independent kinematic modeling technique is also proposed for this system [8].

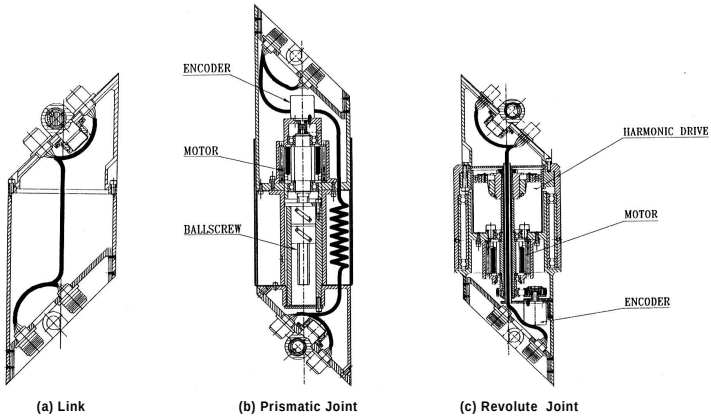


Figure 1.4: Robot modules in the MRS (adapted from [29])

The modular robot at University of Stuttgart [132] consists of link modules of different sizes and a variety of rotational joint modules. The joint modules driven by AC motors in conjunction with differential gears can create complicated 2-DOF or 3-DOF motions. A simple male-female fastening device is designed to connect modules in any order quickly and exactly. All the necessary specifications of link and joint modules are stored in a database file. A program that can select modules to construct special-purpose modular manipulators is proposed.

The TOMMS at Toshiba Corp. [84] contains one type of 1-DOF revolute joint module actuated by a DC motor with a harmonic type reduction gears, and one type of link module. The joint module has three input ports and two output ports. The three input ports are used for the alignment of the rotational axis with three axes of the joint coordinate system respectively, while the two output ports are used to permit the module to either

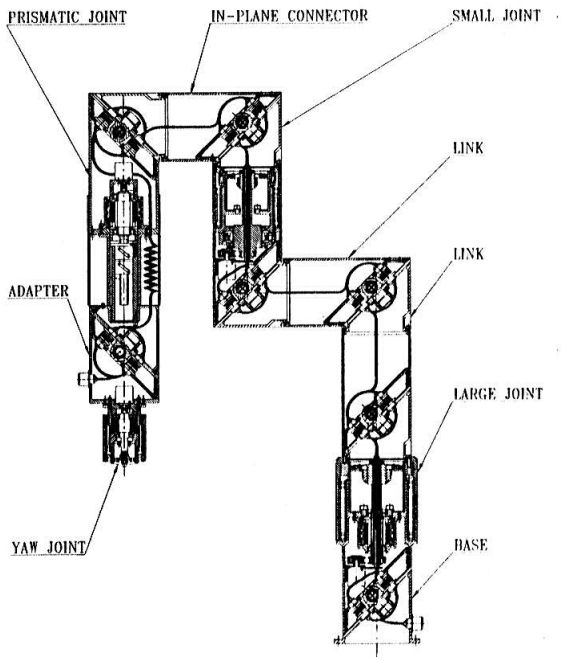


Figure 1.5: A SCARA type modular robot configuration constructed by MRS modules (adapted from [29])

rotate or bend. The link module consists of two cylinders that can slide relative to each other so that the length of the link module is adjustable. Based on numerical inverse kinematics, a position control algorithm is designed for the TOMMS.

An extensive study on modular robot mechanical structures is performed in the University of Texas at Austin [1, 2, 124]. An inventory of joint modules including 1-DOF "elbows", 2-DOF "knuckle", and 3-DOF "shoulders" and "wrists", and a variety of link modules with different lengths, cross-sections, and materials are designed and tested. To evaluate the performance of an individual module, two performance matrices, namely, *joint module performance matrix* and *link module performance matrix* are defined. This research actually advocates a "component" design approach, rather than a general "complete structure" design approach. Several optimization algorithms are also proposed for the simple (one and two-dimensional) robot configuration design [1].

The DRRS at the Science University of Tokyo [42, 44], however, has a very different system design. The idea of this study is to apply the cell system of living creatures to a robot system with some intelligence analogous to the biological gene. There are four types of cells (modules) in the DRRS: joint cells, mobile cells, branching cells, and work

cells. All cells are identical in dimensions. The connection and detachment between cells are carried out by the mobile or joint cells through a hook type coupling mechanism. A cone-shaped mechanism guides the cells during the coupling. The cells in the DRRS possess certain intelligence and communication capability so that they can reconfigure themselves autonomously.

In addition to the research types of modular robot systems, there are a few commercial modular robot systems: RMD-1 by Engineering Services Inc. of Canada and MoRSE by AMTEC GmbH of Germany.

The RMD-1 is a Reconfigurable and Modular (Rotary) Mechatronic Drive system (joint) (Fig. 1.6). Each RMD-1 comprises of a built-in actuator, a gear mechanism, a position encoder, a brake, all of the necessary electronics, and the on-board processing unit. RMD-1 is available in various power capacities starting from 50W to about 400W. The size of RMD-1 varies with its power capacity. Each RMD-1 module can be reconfigured to provide three different types of rotations: roll, pitch, and yaw. Serving as a "building block" of motion control automation, RMD-1 can either be used individually to provide accurate motion control of a payload about a single axis, or as an assembly of several modules to provide control about multiple axes. Because of its inherent modularity, any number of modules can be added or removed easily from an existing configuration. Fig. 1.7 shows two modules interconnected in various configurations. Note that both Fig. 1.6 and Fig. 1.7 are adapted from the product catalog of ESI Company, Canada.

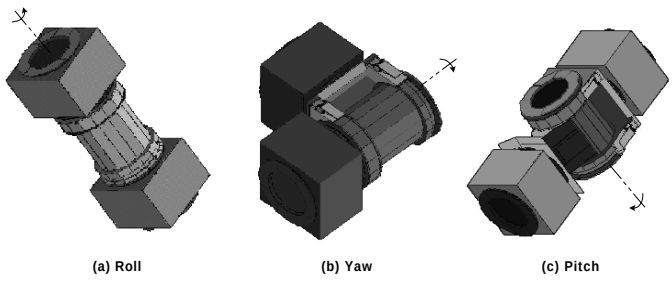


Figure 1.6: Three different types of rotations

The MoRSE system stands for **Mo**dular **R**obot **S**ystem for **E**ngineering. It consists of a variety of drive modules of different sizes: 1-DOF revolute drive modules, 1-DOF linear drive modules, 2-DOF wrist modules, and gripper modules (Fig. 1.8). Each MoRSE mod-

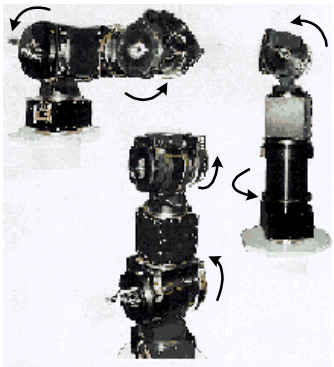


Figure 1.7: Three different (2-DOF) RMD-1 assembly configurations

ule is a totally self-contained unit. The brushless DC servo motor, the motor controller, the harmonic-drive mechanism, the break, the opto-isolated digital and analog I/O, and the fieldbus (CAN bus) interface are all encapsulated in a compact design. Due to the standardized cubic design, the modules of MoRSE system can be connected via simple coupling elements (connectors). An existing robot configuration can be easily altered when required. A PUMA type robot configuration constructed by using MoRSE drive modules is shown in Fig. 1.9. Note that both Fig. 1.8 and Fig. 1.9 are adapted from the product catalog of AMTEC GmbH, Germany.

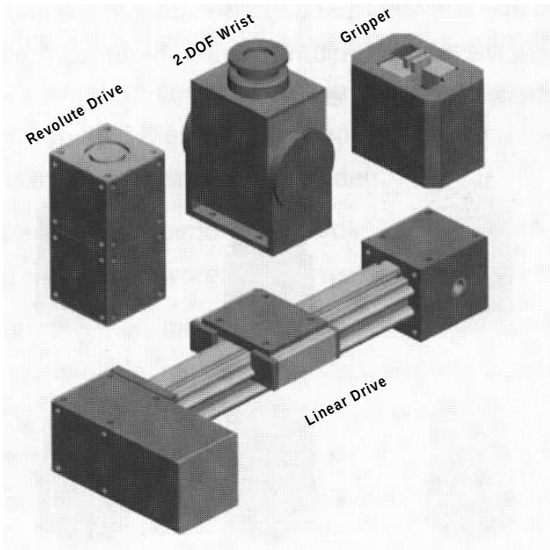


Figure 1.8: MoRSE drive modules

Based on the existing modular robot systems, the conceptual design of a Modular Reconfigurable Robot System (MRRS) has been proposed by us [15]. It contains most of

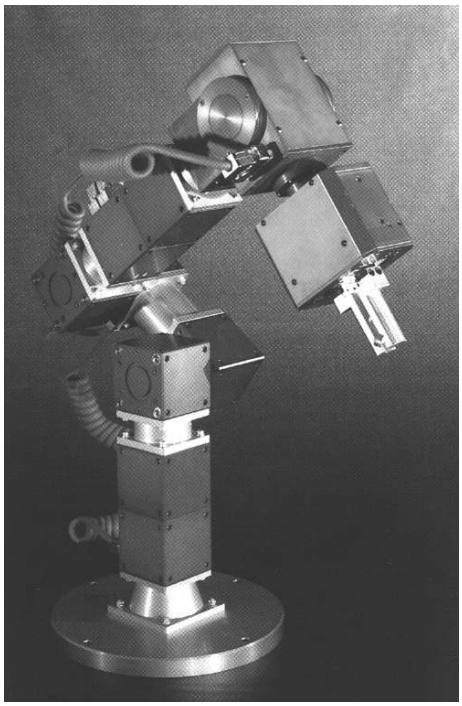


Figure 1.9: A PUMA type robot configuration constructed by MoRSE modules

the features of the existing systems. An individual joint module can be used as an independent motion control element. The fully assembled systems can be used as standard robots with various configurations and *degrees of freedom* (DOFs). The modules making up the MRRS can be divided into two categories, i.e., joint modules and link modules. The joint modules are self-contained control units, including 1-DOF revolute and prismatic joint modules. The connectors for mechanical and electrical coupling of modules have also been designed. Based on the building block principle, each module is designed as a cube with multiple connecting sockets so that many possible configurations can be constructed, e.g., serial and tree-structured types. A detailed introduction to the MRRS is addressed in Chapter 2.

Note that past research efforts on modular robot systems are mainly focused on the hardware design. Various types of robot modules and module connectors have been designed and developed. The software aspect of modular robots, however, is yet to be studied extensively. Algorithms, such as kinematic and dynamic modeling algorithms, employed in most of the existing modular robot systems are transplanted from the conventional robot systems and are configuration-dependent. Module and configuration selection and

optimization algorithms are in their primitive stages. All these factors limit the class of deployable modular robot configurations to a small range. Hence, in order to develop an effective MRRS, the software aspect of modular robots should be emphasized.

1.3 Overview of this Thesis

This thesis addresses several fundamental research topics relating to the MRRS, i.e., conceptual module design, module assembly representation, kinematics, dynamics, calibration, and optimization of robot configurations. The relationships among these research topics are shown in Fig. 1.10. The robot modules are the basic "building blocks" of the system. Module assembly representation enables automatic (kinematic and dynamic) model generation. Kinematics and dynamics algorithms are essential for modular robot configuration optimization, motion planning, and control. Since an MRRS has the capability of forming a large number of robot configurations, developing effective module assembly planning algorithms for specific task requirements is a critical issue for the integration of modular robots. Because of the reconfiguration capability of the MRRS, assembly errors are likely to be introduced to a modular robot. Kinematic calibration, therefore, becomes another important issue as well.

1.3.1 Main Contributions

The main contributions of this work are:

- A set of conceptual robot modules has been designed including revolute joint modules, prismatic joint modules, cubic link modules, and connectors. Each module resembles a cube that has one standard connecting socket on each of its six faces. Due to the cubic and multi-connective design of modules, many possible robot geometries can be constructed, e.g., serial and tree-structured types, for a diversity of task requirements.

- Based on the graph theory, the accessibility matrix and path matrix concepts are employed for indicating the topological structures of modular robots. Algorithms

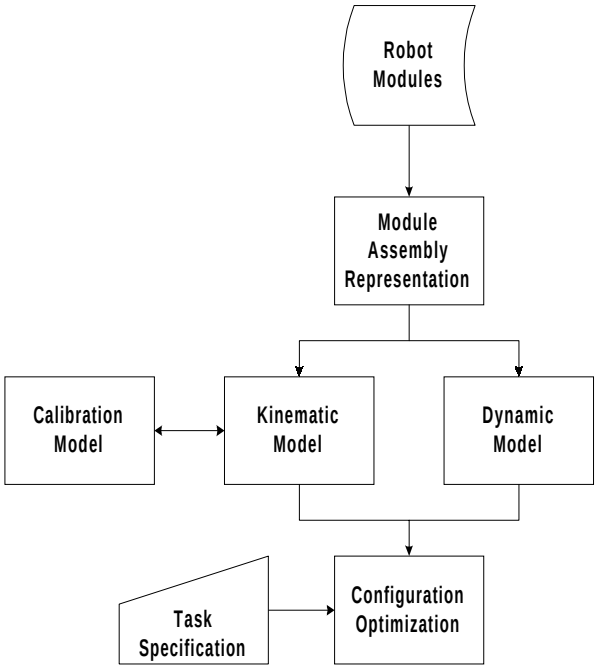


Figure 1.10: Relationships among the research issues

that can automatically derive the accessibility matrix and path matrix from the Assembly Incidence Matrix (AIM) of a modular robot have also been developed. The matrix representation methods pave the way for the development of configuration-independent algorithms.

- Based on the local frame representation of POE formula, both the forward and inverse kinematic algorithms are proposed for general tree-structured modular robots. The forward kinematic algorithm is a recursive method. The inverse kinematic algorithm, on the other hand, follows a numerical approach because of the unlimited configurations of modular robots. By drawing upon the kinematic modeling techniques in geometric robotics, the proposed inverse kinematic model has a very simple and compact form. Instead of the conventional Jacobian matrix, this inverse kinematic algorithm is formulated in terms of the body manipulator Jacobian which can be easily derived by means of linear operation on $se(3)$. The differential change of the end link pose is then formulated as the matrix logarithm on $SE(3)$ accordingly. Besides, both forward and inverse kinematic algorithms are configuration-independent regardless of the types of joints and number of degrees of freedom. Once the AIM of a robot is given, the forward and inverse kinematic models can be generated

automatically.

- Based on the existing techniques in geometric robotics for dynamic modeling, a configuration-independent dynamic formulation algorithm is proposed for general tree-structured modular robots. This algorithm can also generate the dynamic models from a given AIM automatically. The dynamic formulation starts with the modified recursive Newton-Euler algorithm in which the generalized velocities, accelerations, and forces (or wrenches) are expressed in terms of linear operations on $se(3)$. The recursive dynamic model is then formulated into the closed form equations of motion, which are useful for modular robot design, high level manipulation, and motion optimization.

- A configuration-independent kinematic calibration algorithm based on the local POE formula is formulated for general tree-structured modular robots. To a certain extent, the calibration algorithm is a generalization of the numerical inverse kinematics algorithm. The error correction parameters are assumed to be in the relative initial pose of every dyad (two consecutive modules), which are elements of $se(3)$. The gross kinematic errors of a complete robot can be expressed by the matrix logarithm of $SE(3)$, which are also elements of $se(3)$. Therefore, the calibration model can be easily formulated in terms of linear operations on $se(3)$ and provides a clear geometric interpretation of the calibration process.

- An optimization algorithm is proposed for the design of suitable modular robot configurations for specific task requirements. The Minimized Degrees of Freedom (MDOF) concept is introduced into the optimization of robot configurations. Based on the effective coding schemes defined on AIMs, an evolutionary algorithm (EA) approach is employed. To guarantee the operation of EA, a virtual module concept is also introduced into the coding schemes. Significantly, this optimization algorithm can perform both the topological synthesis and dimensional synthesis of modular robot configurations in a uniform way.

In brief, a unified geometric framework that addresses most of the aspects of modular robot problem has been synthesized in this thesis. By taking tools from graph theory and

differential geometry, this geometric framework provides an integrated and user-friendly environment for analysis and design of modular robots. Note that although most of the algorithms developed in this thesis are meant for the MRRS, they are generic and can be applied to all existing modular robot systems as well.

1.3.2 Organization

The remaining chapters of this work are organized as following:

- Chapter 2 introduces the conceptual design of modules.
- Chapter 3 addresses the matrix representation methods for modular robot configurations.
- Chapter 4 addresses kinematic modeling algorithms.
- Chapter 5 addresses dynamic modeling algorithms.
- Chapter 6 addresses the kinematic calibration algorithm.
- Chapter 7 addresses the modular robot configuration optimization algorithm.
- Chapter 8 concludes this work.
- Appendix A presents the computation scheme for the generalized mass matrix of a link assembly.
- Appendix B presents the geometric interpretation of the kinematic errors in a dyad.

Chapter 2

Conceptual Module Design

A major advantage of applying the modular design philosophy to a problem-solving process is that it focuses on a set of basic functional units - the modules. This chapter introduces the conceptual module design which is critical to the performance of the MRRS such as the reconfigurability, the kinematic characteristics, and the dynamic behavior.

The modules making up the MRRS can be divided into two categories, i.e., joint modules and link modules. The joint modules include 1-DOF revolute and prismatic joint modules. A modular robot is then defined as an assembly of these modules with a unique sequence. Each type of module has a cubic design and is made in two different sizes for a diversity of task requirements. The design of connectors for connecting two modules mechanically and electrically is also presented.

This chapter is organized as follows. Section 2.1 discusses the basic requirements of module design. The design of joint modules, link modules, and connectors are introduced in Sections 2.2, 2.3, and 2.4 respectively.

2.1 Module Design Requirements

In order to develop an MRRS with high reliability and reconfigurability, the module design should satisfy the following requirements:

- Each module should be a self-contained independent unit.

- Each module should be designed with symmetric geometry and multi-connectivity.
- Each module should be connectable to any other modules regardless of their types and sizes.
- Each module should be designed with minimum weight and inertia.
- The connections between modules should be accurate and rigid.

The first requirement focuses on the reconfigurability of the system. This implies that the entire joint actuator must be packed within the joint module. The second and third requirements are the essential embodiment to increase the reconfigurability of the system. When each type of module is designed as a symmetric object with multiple connecting sockets, many possible robot configurations including both serial and tree-structured types can be constructed. Owing to the first requirement, a modular robot yields geometries with most of actuators being located on the moving arm itself. Hence, the fourth requirement is important for a modular robot to increase its working capability such as the maximum payload and the operation speed. In addition, the last requirement is essential for a modular robot to perform tasks requiring high accuracy and repeatability.

2.2 Joint Modules

There are five types of lower pair joints commonly used in robot systems: revolute (1-DOF), prismatic (1-DOF), cylindrical (2-DOF), planar (2-DOF), and spherical (3-DOF) [14]. Since a multi-DOF joint can be obtained by combining several 1-DOF joints, it is unnecessary to have such a complex multi-DOF joint design in the MRRS. In the MRRS, only two types of 1-DOF joints: revolute joints and prismatic joints are considered.

2.2.1 Revolute Joint Modules

A revolute joint module is designed to generate rotary motion. In the MRRS, two different sizes of revolute joint modules that have the same mechanical structure are considered. As

shown in Fig. 2.1, the revolute joint module is designed as a cube. The actuator is packed within the cubic housing. There is one standard connecting socket for both mechanical and electrical connections on each of the six faces of the cubic module. Among the six connecting sockets, one is the revolute socket driven by the built-in actuator. In order to get a high torque/weight ratio within a compact space, the commercial FH-2000 series AC hollow-shaft servo actuators (from Harmonic Drives GmbH, Germany) are selected for the design, as shown in Fig. 2.2. Such an actuator combines a maintenance-free AC servo motor with an incremental encoder, a backlash-free harmonic drive, and a brake in a compact design. The unique feature of the new series of actuators is the hollow shaft which can allow the electrical wiring to pass through the shaft.

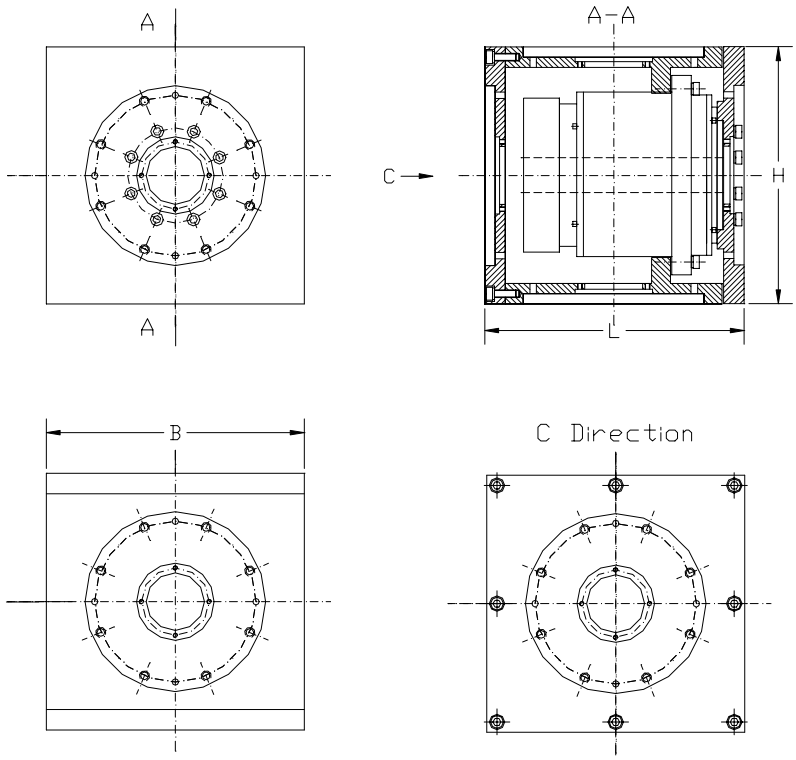


Figure 2.1: Revolute joint module

The connecting socket is a circular machined hole on the cubic module housing for matching with the flange of the connector unit. In order to obtain a good axial alignment between connected modules, the hole's inner cylindrical surface is machined with high precision. At the bottom face of the connecting socket, eight bolt holes and four locating holes (smaller ones) are drilled on the same pitch circle for rotational alignment and fastening respectively. Note that only one locating hole will be used for the assembly action

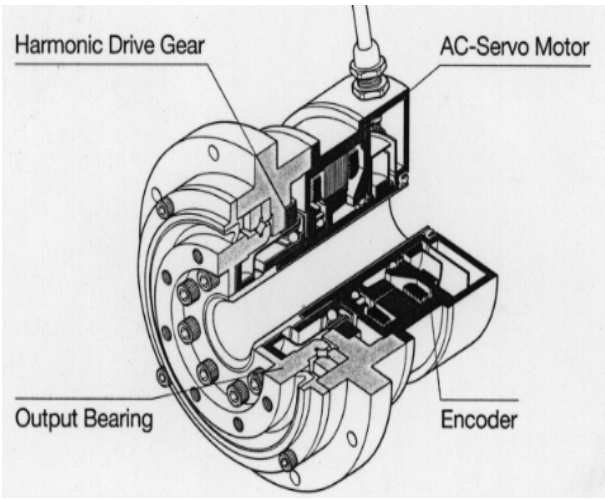


Figure 2.2: FH-2000 series motor

between a module and a connector. Hence, the connector can be attached to a particular socket of the module in four different orientations.

The technical specifications of revolute joint modules are listed in Table 2.1. Note that the term 'CM' refers to the *center of mass*. The location of CM is expressed in the module frame which is located at the center of the cube as shown in Fig. 2.3. The initial matrix is given with respect to the frame at CM, termed as CM frame. The CM is parallel with the module frame.

Table 2.1: Technical specifications of revolute joint modules

Item	Small Modules	Large Modules
Dimensions(m) $L \times B \times H$	$0.2 \times 0.2 \times 0.2$	$0.275 \times 0.275 \times 0.275$
Output Speed (rpm): Rated/Max.	30/40	25/35
Output Torque (Nm): Rated/Max.	51/157	137/570
Weight (kg)	7.0	20.5
Motor Type	FHA25B-3015	FHA40A-2536
Motor Power (W)	160	360
Encoder Resolution (ppr)	1500	1500
Position of CM (m)	$(0, 0, 0.027)^T$	$(0, 0, 0.044)^T$
Inertial Matrix ($kg.m^2$)	$diag[0.0313, 0.0313, 0.0306]^T$	$diag[0.173, 0.173, 0.165]^T$

2.2.2 Prismatic Joint Modules

A prismatic joint module is designed to generate translational motion. In the MRRS, two sizes of prismatic joint modules that have the same mechanical structure are considered as well. As shown in Fig. 2.4, the prismatic joint module is designed as a combination of

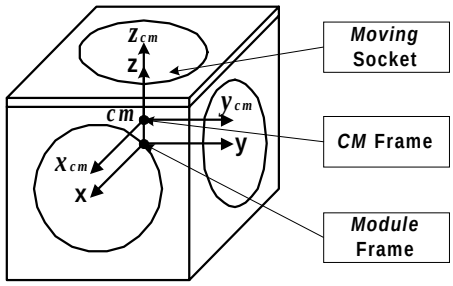


Figure 2.3: Coordinate frames in a revolute joint module

a cube and a cylinder. The translational motion is realized by the cylinder's "telescopic" movement. Such a module also has six standard connecting sockets on its housing. One of them is the translational socket driven by an actuator in conjunction with a ball-screw transmission mechanism. We choose FH series AC servo motors (without harmonic drives) as the actuators of the prismatic joint modules for the same reason mentioned above. The design of the connecting socket is the same as the revolute joint modules.

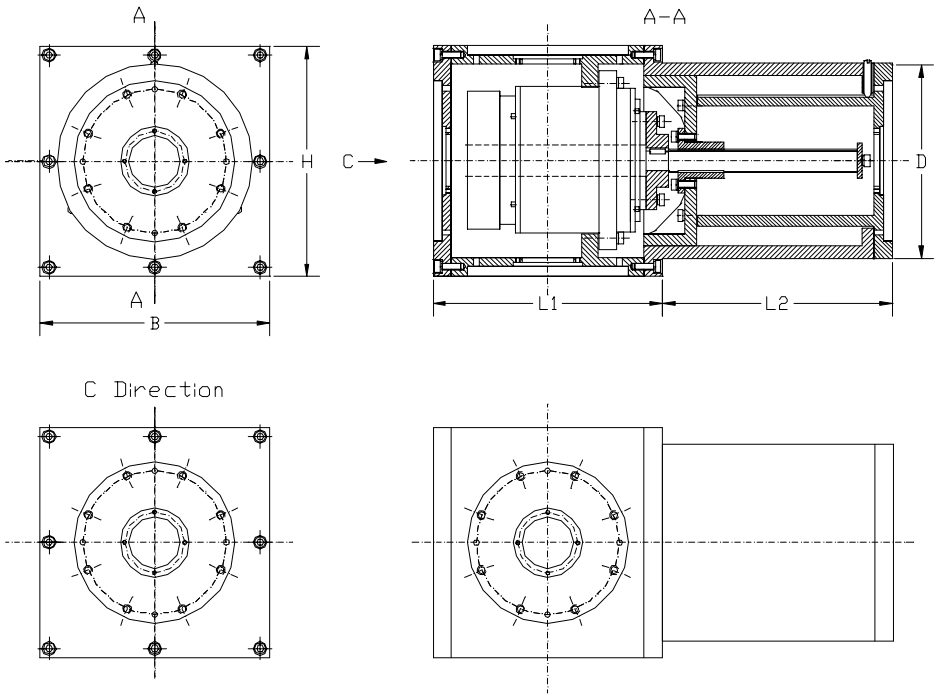


Figure 2.4: Prismatic joint module

The technical specifications of the prismatic joint modules are listed in Table 2.2. The location of CM is expressed in the module frame which is located at the center of the cube as shown in Fig. 2.5. The initial matrix is given with respect the CM frame. Note that for a prismatic joint module, the linear motion of the cylindrical tube will change

the position of the CM (with respect to the module frame). In this case, the whole module cannot be considered as a rigid body. Hence, we divide a prismatic joint module into two parts: the movable cylindrical tube and the remaining part which, without confusing, is still called a prismatic joint module. The cylindrical tube that is rigidly connected with the approaching connector/adapter is considered as the extended part of the connector/adapter. The mass and inertial matrix listed in Table 2.2 do not include those of the cylindrical tube.

Table 2.2: Technical specifications of prismatic joint modules

Technical specification	Small Modules	Large Modules
Dimensions(m) $L1 \times B \times H/L2 \times D$	$0.2 \times 0.2 \times 0.2/0.2 \times 0.175$	$0.275 \times 0.275 \times 0.275/0.275 \times 0.25$
Output Speed (m/sec): Rated/Max.	0.2/0.267	0.208/0.292
Output Force (N): Rated/Max.	800/2463	1703/7085
Working Stork (m)	0.1	0.15
Weight (kg)	8.5	24
Motor Type (without harmonic drive mechanism)	FHA25B-3015	FHA40A-2536
Motor Power (W)	160	360
Encoder Resolution (ppr)	1500	1500
Ball-Screw Mechanism	TMF20-OH	TMF25-OH
Position of CM (m)	$(0, 0, 0.050)^T$	$(0, 0, 0.070)^T$
Inertial Matrix ($kg.m^2$)	$diag[0.0592, 0.0592, 0.0434]^T$	$diag[0.267, 0.267, 0.222]^T$

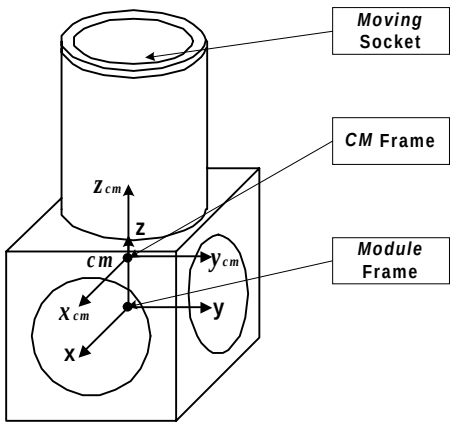


Figure 2.5: Coordinate frames in a prismatic joint module

2.3 Link Modules

Link modules are used to connect or support joint modules by means of connectors. The geometry of the link module is similar to the revolute joint module but without the built-in actuator and the moving socket. The six connecting sockets in a link module are all fixed

Table 2.3: Technical specifications of cubic link modules

Item	Small Module	Large Module
Dimensions(m) $L \times B \times H$	$0.2 \times 0.2 \times 0.2$	$0.275 \times 0.275 \times 0.275$
Weight (kg)	2.1	5.2
Position of CM (m)	$(0, 0, 0)^T$	$(0, 0, 0)^T$
Inertial Matrix ($kg.m^2$)	$diag[0.025, 0.025, 0.025]^T$	$diag[0.115, 0.115, 0.115]^T$

as shown in Fig, 2.6. Also, two different sizes of link modules are designed for MRRS. By connecting several link modules with connectors, we can create links with various shapes and dimensions.

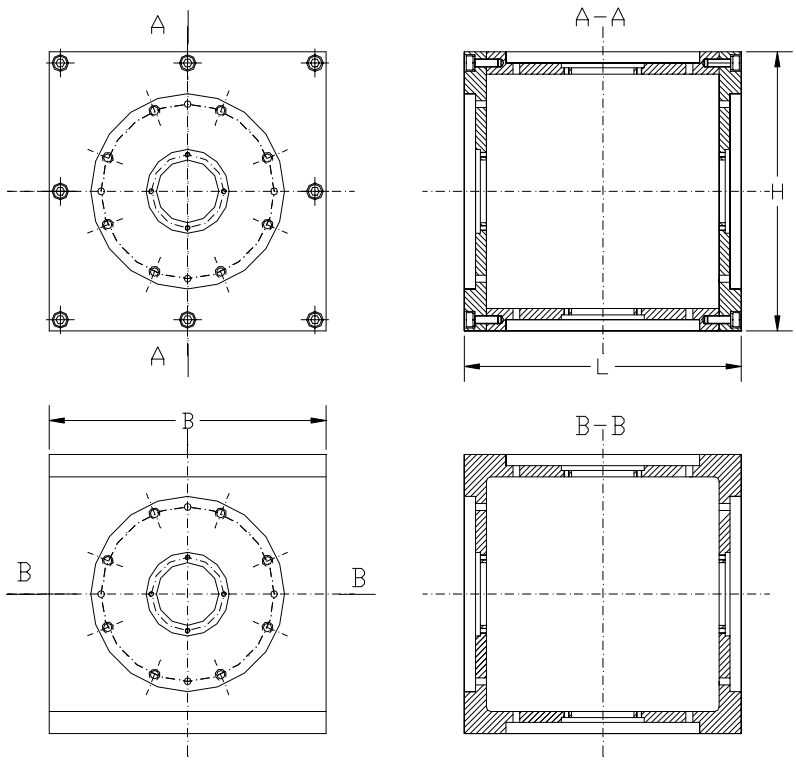


Figure 2.6: Cubic link module

The technical specifications of link modules are listed in Table 2.3. The location of CM is expressed in the module frame which is located at the center of the cube as shown in Fig. 2.7. The initial matrix is given with respect to the CM frame. Because the link module is a symmetric cube, the model frame and the CM frame coincide with each other.

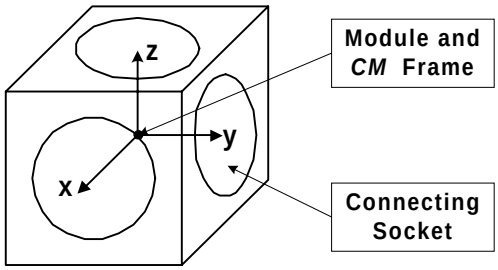


Figure 2.7: Coordinate frames in cubic link module

2.4 Connectors

The connection between two modules is through a connector. The connection between the module and the connector is through the flange. Since the sizes of connecting sockets are different for different series of modules, three types of connectors are designed, namely the large one (to connect two large modules), the small one (to connect two small modules), and the medium one (to connect two modules of different sizes). The large and small connectors have the same geometry (Fig 2.8(a)), while the medium one, also called an adapter, has a different design (Fig 2.8(b)).

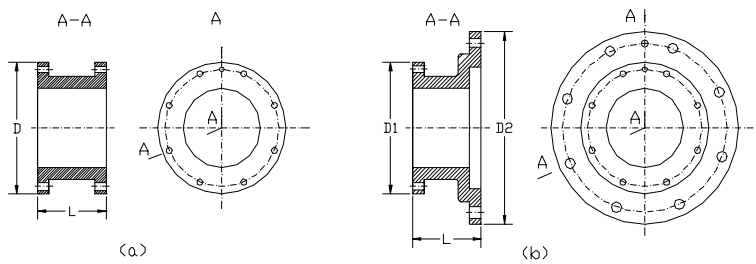


Figure 2.8: Connector/adapter

The flange of the connector to be matched with the inner cylindrical surface of the connecting socket of a module, should be accurately machined in order to get the required axial alignment. On the flange, there are eight holes arranged in a circle for fastening purpose. An additional hole is for rotational alignment and positioning, so that as mentioned before, a connector can be attached to a connecting socket in four different orientations. The connector has a hollow cylindrical structure so that the electrical connector can be installed inside.

The technical specifications of the connectors are listed in Table 2.4. Note that when

a connector or an adapter is connected with the output socket (moving socket) of a prismatic joint module, the moving part will be considered as the extended part of the connector/adapter. Hence, the mass, the position of CM, and the initial matrix of the newly defined connectors/adapter will be different from the original ones, as listed in Table 2.5. The coordinate frames in a connector or an adapter is shown in Fig. 2.7

Table 2.4: Technical specifications of connectors/adapter

Technical specification	Small Connector	Adapter	Large Connector
Dimensions(m) $D1 \times L \times D2$	$\phi 0.15 \times 0.075 \times \phi 0.15$	$\phi 0.15 \times 0.075 \times 0.2$	$\phi 0.2 \times 0.075 \times \phi 0.2$
Weight (kg)	0.45	0.6	0.85
Position of CM (m)	$(0, 0, 0)^T$	$(0, 0, -0.007)^T$	$(0, 0, 0)^T$
Inertia ($kg.m^2$)	$diag[0.0004, 0.0004, 0.0081]^T$	$diag[0.0003, 0.0003, 0.0045]^T$	$diag[0.0004, 0.0004, 0.0081]^T$

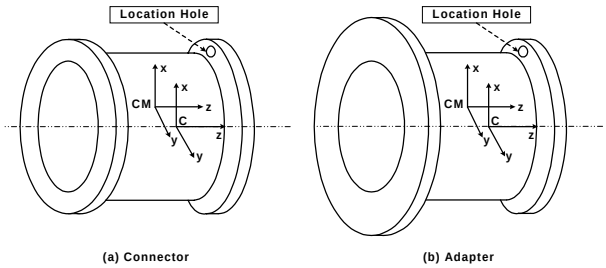
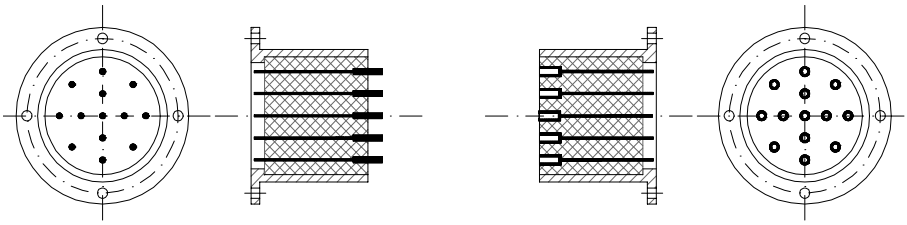


Figure 2.9: Coordinate system in a connector/adapter

In order to achieve electrical connection, a standard male and female electric connectors as shown in Fig. 2.10, are designed. The electric connectors will be fastened onto a module's connecting socket through four bolts on the inner circle. The electric connectors are used in pairs. For example, if a male electric connector is installed on a connecting socket of a module, a female one must be installed on the corresponding socket of the joining module. Note that, all the male or female electric connectors used in the MRRS are the same. When two modules are connected through a mechanical connector, the electrical connectors inside will also be connected. All communication signals and power supplies for the modules will then pass through the electrical connection, and no external wiring is needed.

Table 2.5: Technical specifications of connectors/adapter (including the moving part of the prismatic joint module)

Item	Small Connector	Adapter	Large Connector
Weight (kg)	1.95	3.75	4.0
Position of CM (m)	$(0, 0, -0.065)^T$	$(0, 0, -0.108)^T$	$(0, 0, -0.087)^T$
Inertia($kg.m^2$)	$diag[0.0110, 0.0110, 0.0065]^T$	$diag[0.0317, 0.0317, 0.0546]^T$	$diag[0.0377, 0.0377, 0.0582]^T$



(a) Male **(b) Female**

Figure 2.10: Electronic connector

2.5 Discussion

In this chapter, the mechanical design of the joint and link modules is addressed. The individual modules considered include revolute joint modules, prismatic joint modules, and cubic link modules. An individual joint module can be used as an independent motion control element, and the fully assembled systems can be used as standard robots with various configurations and *degrees of freedom* (DOFs). In order to broaden the application area of the modular robot system, each type of module comes with two sizes. Three sizes of mechanical connectors and a set of standard electrical connectors have also been designed for mechanical and electrical connections. Based on the building block principle, each module is designed as a cube with multiple connecting sockets so that many possible configurations can be constructed with a small inventory of modules. Note that the modules have not yet been fabricated because the electronic design is still under research. However, The algorithms developed in this thesis follow a general framework, and can be applied to other modular robot systems as well.

Chapter 3

Modular Robot Representation

In the previous chapter, the conceptual module designs are presented. With an inventory of such modules, one can create many robot configurations for different applications. This chapter addresses the issues pertinent to the mathematical representation of modular robot assembly configurations, which are the basis of the automatic model generation techniques to be discussed in the following chapters.

The fundamental work in the representation of modular robot configurations has been well established by Chen [14], in which a formal definition of the Assembly Incidence Matrix (AIM) is proposed based on the graph theory. The AIM has a very compact form and contains all the necessary information regarding the modular robot assembly configurations such as the types, the connecting sequence, and the relative assembly orientations of modules. Given an AIM, a robot configuration can be uniquely constructed. Following this approach, a modified AIM representation scheme is proposed for the MRRS that we are working on [17]. The accessibility matrix and path matrix concepts are employed for indicating the topological structures of a general tree-structured modular robot, which can facilitate the automatic model generation techniques. Algorithms that can automatically derive the accessibility matrix and the path matrix from the Assembly Incidence Matrix (AIM) of a modular robot have also been discussed.

This chapter is organized as follows. Section 3.1 reviews some basic concepts, definitions, and algorithms in graph theory. Section 3.2 discusses issues of the kinematic graph and its applications on modular robot system. Based on the module designs, the re-classification

of links and joints is addressed in Section 3.3. The modified Assembly Incidence Matrix (AIM) is finally introduced in Section 3.4.

3.1 Graphs

In this section, several basic concepts and definitions in graph theory are reviewed. All the graphs discussed here are simple graphs. For more details, please refer to [10, 129].

3.1.1 Basic Graph Definitions

Definition 1 (Graph) *A graph $G = (V, E)$ consists of a vertex set, $V(G)$, and an edge set, $E(G)$, such that every edge in $E(G)$ is associated with a pair of vertices in $V(G)$ (Fig. 3.1(a)).*

Definition 2 (Directed Graph) *A directed graph (or digraph) is a graph in which every edge is associated with an ordered pair of vertices (Fig. 3.1(b)).*

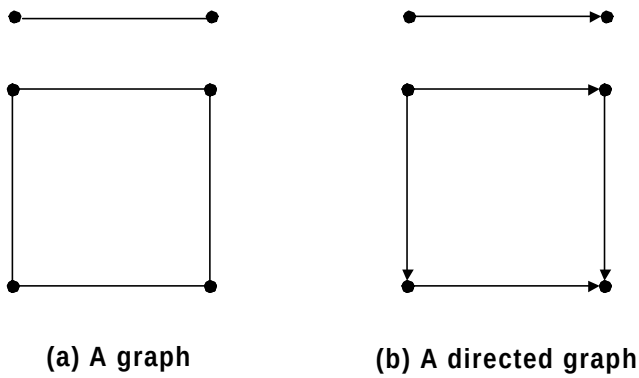
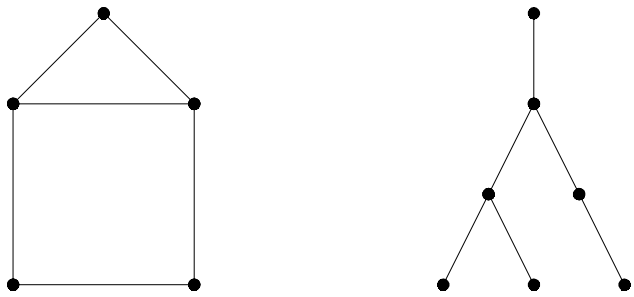


Figure 3.1: A graph and a directed graph

Definition 3 (Connected Graph) *A graph is connected if for every pair of distinct vertices $\{\mathrm{v_i}, \mathrm{v_j}\}$ there is a path from $\mathrm{v_i}$ to $\mathrm{v_j}$ (Fig. 3.2(a)).*

Definition 4 (Tree) *A tree is a (minimal) connected graph without cycle (Fig. 3.2(b)).*



(a) A connected graph (a) A tree

Figure 3.2: A connected graph and a tree

Note that, for a tree, if the root is given, the tree will become a directed or rooted tree. Moreover, a tree with n vertices has $n-1$ edges.

Definition 5 (Labeled Graph) *A labeled graph is a graph in which the vertices are labeled by* $\mathrm{v}_1,\mathrm{v}_2,\ldots,\mathrm{v}_\mathrm{m}$*, and the edges are labeled by* $\mathrm{e}_1,\mathrm{e}_2,\ldots,\mathrm{e}_\mathrm{n}$*, such that* $V=\{\mathrm{v}_1,\mathrm{v}_2,\ldots,\mathrm{v}_\mathrm{m}\}$ *and* $E=\{\mathrm{e}_1,\mathrm{e}_2,\ldots,\mathrm{e}_\mathrm{n}\}$ *(Fig. 3.3(a)).*

Usually, we do not differentiate among the types of vertices (edges). If the vertices (edges) are assumed to be different, a more complex graph, termed specialized graph, should be defined.

Definition 6 (Specialized Graph [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* $\tilde{G}=(G,\tilde{V},\tilde{E},f_v,f_e)$*, where* $\tilde{V}(\tilde{E})$ *represents the set of vertex (edge) types on* G*, and* $f_v:V\mapsto\tilde{V}$ *and* $f_e:E\mapsto\tilde{E}$ *are injective mappings which are called vertex and edge assignments respectively (Fig. 3.3(b)).*

Example 3.1: A labeled graph G is shown in Fig. 3.3(a), where $V=(\mathrm{v}_1,\mathrm{v}_2,\ldots,\mathrm{v}_6)$ and $E=(\mathrm{e}_1,\mathrm{e}_2,\ldots,\mathrm{e}_5)$. Its specialized graph $\tilde{G}$ is shown in Fig. 3.3(b), where $\tilde{V}=\{$A, B, C$\}$ and $\tilde{E}=\{$x, y, z$\}$.

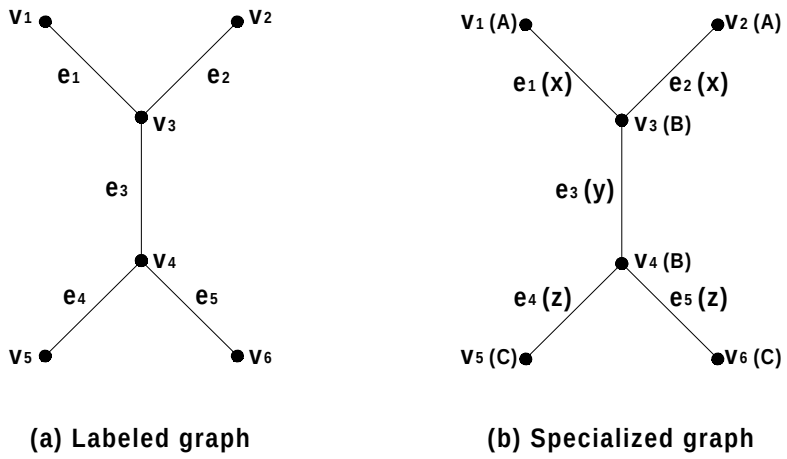


Figure 3.3: A labeled graph and a specialized graph

3.1.2 Matrix Representation of Graphs

A graph can also be expressed in matrix forms that are very convenient for computer processing. The following matrix representations for graphs are extensively used throughout this thesis.

Definition 7 (Adjacency Matrix) *Let $G = (V, E)$ be a labeled graph, where $V = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_\mathrm{m}\}$ and $E = \{\mathrm{e}_1, \mathrm{e}_2, \ldots, \mathrm{e}_\mathrm{n}\}$. The adjacency matrix $\mathcal{B}(G)$ is the $m \times m$ matrix in which the entry in row i and column j is 1, if there is an edge joining vertices v_i and v_j; it is 0, otherwise.*

Definition 8 (Incidence Matrix) *Let $G = (V, E)$ be a labeled graph, where $V = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_\mathrm{m}\}$ and $E = \{\mathrm{e}_1, \mathrm{e}_2, \ldots, \mathrm{e}_\mathrm{n}\}$. The incidence matrix $\mathcal{M}(G)$ is the $m \times n$ matrix in which the entry in row i and column j is 1, if edge e_j is incident on vertex v_i ; it is 0, otherwise.*

Example 3.2: The adjacency and incidence matrix of the underlying graph shown in Fig. 3.4(a) can be written as:

$$\mathcal{B}(G) = \begin{array}{c} \\ \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \\ \mathbf{v}_5 \\ \mathbf{v}_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathbf{v}_0 & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 & \mathbf{v}_5 & \mathbf{v}_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$\mathcal{M}(G) = \begin{array}{c} \\ \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \\ \mathbf{v}_5 \\ \mathbf{v}_6 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 & \mathbf{e}_4 & \mathbf{e}_5 & \mathbf{e}_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

According to the definition of adjacency matrix of a underlying graph, $\mathcal{B}(G)$ is a square and symmetric matrix.

Definition 9 (Adjacency Matrix - for a directed graph) *Let $\vec{G} = (V, E)$ be a labeled digraph graph, where $V = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_\mathrm{m}\}$ and $E = \{\mathrm{e}_1, \mathrm{e}_2, \ldots, \mathrm{e}_\mathrm{n}\}$. The adjacency matrix $\mathcal{B}(\vec{G})$ is the $m \times m$ matrix in which the entry in row i and column j is 1, if there is an edge from vertices v_i to v_j; it is 0, otherwise.*

Definition 10 (Incidence Matrix - for a directed graph) *Let $\vec{G} = (V, E)$ be a labeled digraph graph, where $V = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_\mathrm{m}\}$ and $E = \{\mathrm{e}_1, \mathrm{e}_2, \ldots, \mathrm{e}_\mathrm{n}\}$. The incidence matrix $\mathcal{M}(\vec{G})$ is the $m \times n$ matrix in which the entry in row i and column j is 1 if v_i is the initial vertex of e_j; it is -1, if v_i is the terminal vertex of e_j; it is 0, otherwise.*

Example 3.3: The adjacency and incidence matrix of the labeled digraph shown in Fig. 3.4(b) can be written as:

$$\mathcal{B}(\vec{G}) = \begin{array}{c} \\ \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \\ \mathbf{v}_5 \\ \mathbf{v}_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathbf{v}_0 & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 & \mathbf{v}_5 & \mathbf{v}_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$\mathcal{M}(\vec{G}) = \begin{array}{c} \\ \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \\ \mathbf{v}_5 \\ \mathbf{v}_6 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 & \mathbf{e}_4 & \mathbf{e}_5 & \mathbf{e}_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \end{array}$$

It is not difficult to see that the structure of a labeled graph/digraph can be described by its adjacency or incidence matrix. The incidence matrix, in general, is not a square matrix so that it lacks some of the power of the *adjacency matrix* in computation aspect. Despite this shortcoming, the *incidence matrix* has some valuable uses. For example, the incidence relationship among all vertices and edges of a graph can be fully described by such a vertex-edge incidence matrix.

If $\vec{G}$ is a labeled digraph derived from a labeled graph G (as shown in Fig. 3.4), there is a simple connection between the adjacency matrix of G, $\mathcal{B}(G)$, and the incidence matrix of $\vec{G}$, $\mathcal{M}(\vec{G})$, such that:

$$\mathcal{B}(G) = \mathcal{D}(\vec{G}) - M(\vec{G})M(\vec{G})^T, \tag{3.1}$$

where $\mathcal{D}(\vec{G})$ is the $n \times n$ diagonal matrix in which the entry in row i and column i is equal to the degrees of vertex $\mathrm{v_i}$ in $\vec{G}$, i.e., the number of nonzero entries in the ith row of $\mathcal{M}(\vec{G})$.

Definition 11 (Accessibility Matrix) *Let $\vec{G} = (V, E)$ be a labeled digraph, where $V = \{\mathrm{v_1}, \mathrm{v_2}, \ldots, \mathrm{v_m}\}$ and $E = \{\mathrm{e_1}, \mathrm{e_2}, \ldots, \mathrm{e_n}\}$. The accessibility matrix $\mathcal{R}(\vec{G})$ is the $m \times m$*

matrix in which the entry in row i and column j is 1, if there is at least one path from $\mathrm{v_i}$ *to* $\mathrm{v_j}$*; it is 0, otherwise.*

Example 3.4: The accessibility matrix of the directed graph $\vec{G}$ as shown in Fig. 3.4(b) can be written as

$$\mathcal{R}(\vec{G}) = \begin{array}{c} \\ \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \\ \mathbf{v}_5 \\ \mathbf{v}_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathbf{v}_0 & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 & \mathbf{v}_5 & \mathbf{v}_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

The accessibility matrix $\mathcal{R}(\vec{G})$ can be simply derived from the powers of adjacency matrix $\mathcal{B}(\vec{G})$ according to the following computation procedure:

1. Let $\mathcal{Q}(\vec{G}) = \mathcal{B}(\vec{G}) + \mathcal{B}^2(\vec{G}) + \ldots + \mathcal{B}^m(\vec{G})$
2. Convert $\mathcal{Q}(\vec{G})$ into $\mathcal{R}(\vec{G})$ such that if $q_{ij} \neq 0$, then $r_{ij} = 1$; it is 0, otherwise.

Definition 12 (Path Matrix) *Let $\vec{G} = (V, E)$ be a directed tree with m pendant vertices, where $V = \{\mathrm{v_0}, \mathrm{v_1}, \ldots, \mathrm{v_n}\}$ and $E = \{\mathrm{e_1}, \mathrm{e_2}, \ldots, \mathrm{e_n}\}$. The path matrix $\mathcal{P}(\vec{G})$ is the $m \times (n+1)$ matrix in which the entry in row i and column j is 1, if path i contains $\mathrm{v_{j-1}}$ $(i = 1, 2, \ldots, m; j = 1, 2, \ldots, n+1)$; it is 0, otherwise.*

Example 3.5: The path matrix of the directed (rooted) tree shown in Fig. 3.4(b) can be written as

$$\mathcal{P}(\vec{G}) = \begin{array}{c} \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathbf{v}_0 & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 & \mathbf{v}_5 & \mathbf{v}_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

The path matrix of a directed (rooted) tree $\mathcal{P}(\vec{G})$ can be derived from its accessibility matrix $\mathcal{R}(\vec{G})$ according to the following computation scheme:

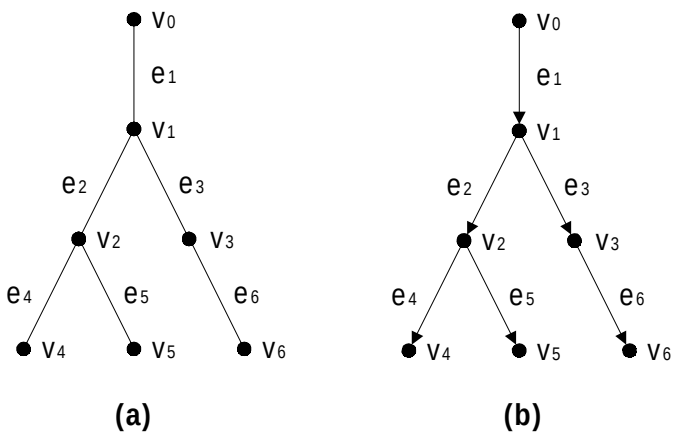


Figure 3.4: A tree and a directed tree

1. Let $\mathcal{U}(\vec{G}) = \mathcal{R}(\vec{G}) + \mathcal{I}_{n\times n}$, where $\mathcal{I}_{n\times n}$ is a $n \times n$ identity matrix in which n is the number of vertices.

2. Delete column i of $\mathcal{U}(\vec{G})$, if the entries in row i of $\mathcal{R}(\vec{G})$ are not all equal to 0.

3. $\mathcal{P}(\vec{G})$ is given by the transpose of the resulting $\mathcal{U}(\vec{G})$

A tree G becomes a rooted or directed tree $\vec{G}$ when the root vertex is specified. If the incidence matrix of a underlying tree, $\mathcal{M}(G)$, and the root vertex are given, we can derive various matrix representations of $\vec{G}$, such as $\mathcal{M}(\vec{G})$, $\mathcal{B}(\vec{G})$, $\mathcal{R}(\vec{G})$, and $\mathcal{P}(\vec{G})$, by employing a tree traversing algorithm, e.g., *Breadth-first search* (BFS) algorithm or *Depth-first search* (DFS) [13, 116] algorithm, to search $\mathcal{M}(G)$. Alternatively, if the rows (vertices) in $\mathcal{M}(\vec{G})$ are arranged in the BFS searching sequence as shown in Fig. 3.4a and Example 3.2 (refer to [13] for more details), then $\mathcal{M}(\vec{G})$, $\mathcal{B}(\vec{G})$, $\mathcal{R}(\vec{G})$, and $\mathcal{P}(\vec{G})$ can be derived by the following computation procedure:

1. Initialization: Given $\mathcal{M}(G)$;

2. Set $\mathcal{M}(\vec{G})$ equal to $\mathcal{M}(G)$ and then let the first row remain unchanged and the first nonzero entry in each of the remaining rows (from left to right) become -1;

3. Set the entry in row i and column i of $\mathcal{D}(\vec{G})$ to be equal to the number of nonzero entries in i^{th} row of $\mathcal{M}(G)$ or $\mathcal{M}(\vec{G})$;

4. Compute $\mathcal{B}(G)$ by Equation (3.1);
5. Set $\mathcal{B}(\vec{G})$ be equal to $\mathcal{B}(G)$ and then let all the entries in the lower triangular block become zero;
6. Compute $\mathcal{R}(\vec{G})$ according the powers of $\mathcal{B}(\vec{G})$;
7. Compute $\mathcal{P}(\vec{G})$ from $\mathcal{R}(\vec{G})$ by the algorithm proposed previously.

A specialized graph mentioned in the previous section can also be described in an incidence-matrix-like structure as well. An additional row and column are required for the assignments of vertex types and edge types respectively. Let $\tilde{G}$ be a specialized graph with m vertices and n edges, the following matrix representation is defined for $\tilde{G}$.

Definition 13 (Extended Incidence Matrix [14]) *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:*

- $m_{ij} = 1$, *if* $\mathrm{e_j}$ *is incident on* $\mathrm{v_i}$; $m_{ij} = 0$, *otherwise* $(i = 1,\ldots,m;\ j = 1,\ldots,n)$.
- $m_{i,n+1} = f_v(\mathrm{v_i}) \in \tilde{V}$, *which is* $\mathrm{v_i}$*'s vertex assignment* $(i = 1,\ldots,m)$.
- $m_{m+1,j} = f_e(\mathrm{e_j}) \in \tilde{E}$, *which is* $\mathrm{e_j}$*'s eage assignment* $(j = 1,\ldots,n)$.
- $m_{m+1,n+1} = 0$.

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

Example 3.6 : The eIM of the specialized graph shown in Fig. 3.3(b) is given by

$$\mathcal{M}(\tilde{G}) = \begin{array}{c} \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \\ \mathbf{v}_5 \\ \mathbf{v}_6 \\ \tilde{\mathbf{E}} \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 & \mathbf{e}_4 & \mathbf{e}_5 & \tilde{\mathbf{V}} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & A \\ 0 & 1 & 0 & 0 & 0 & A \\ 1 & 1 & 1 & 0 & 0 & B \\ 0 & 0 & 1 & 1 & 1 & B \\ 0 & 0 & 0 & 1 & 0 & C \\ 0 & 0 & 0 & 0 & 1 & C \\ \mathrm{x} & \mathrm{x} & \mathrm{y} & \mathrm{z} & \mathrm{z} & 0 \end{pmatrix} \end{array}$$

3.2 Kinematic Graphs

In mechanism design, kinematic chains of links and joints are often represented by graphs. As a joint can be connected to two links only and a link may accept more than one joint, one logical way to convert a kinematic chain to a graph is to replace the joints by edges and links by vertices. Such a graph representation is called a kinematic graph [35] and has been widely used in mechanism synthesis and design [14, 52, 130, 133]. Fig. 3.5 shows a planar 6-bar linkage and its kinematic graph. Note that the ground is treated as a fixed link here.

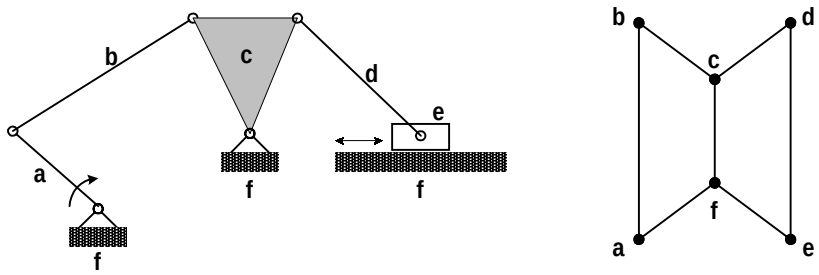


Figure 3.5: A planar 6-bar linkage and its kinematic graph

A modular robot is a collection of link and joint modules so that it is essentially a kinematic chain of links and joints. It follows that a modular robot after conversion can also be represented by a kinematic graph. This graph technique provides a framework for the study of underlying geometric and kinematic properties of a modular robot, for example, the functional difference between a star type graph topology (a walking robot) and a serial type graph topology (an industrial robot). Section 3.4 will show that a modular robot assembly configuration can be fully described by adding more structural and connecting information of modules to the incidence matrix of a kinematic graph. The graph theory and numerous graph algorithms thus can be directly employed for enumeration and classification of modular robot assembly configurations [14].

3.3 Re-classification of Links and Joints

Based on the module designs, any two modules in the MRRS, regardless of their types and sizes, can be connected through a connector. For instance, two joint modules can also be assembled together. Therefore, we cannot directly convert such a modular robot into a kinematic graph just by replacing the joint modules by edges and link modules by vertices. To solve this problem, a re-classification scheme for links and joints is proposed to translate a modular robot into a common kinematic chain.

In this re-classification scheme, each module, regardless of its type, is considered as a "link"; each connector is considered as a "joint" between two consecutive modules. The connecting points on the modules are termed *connecting sockets* or *connecting ports*. The connector is treated as a "revolute joint" when fastened to the rotating socket of the revolute joint module; a "prismatic joint" when fastened to the translating socket of the prismatic joint module; and a "fixed joint" when fastened to two fixed sockets of any modules. The fixed joint does not allow any joint displacement, i.e., the two modules are rigidly connected together. Note that, two moving sockets of different modules cannot be connected through a connector in the MRRS.

According to the dimensions of the actually designed modules and the re-classification scheme, the link set, L, consists of six elements: $L = \{L_{c1}, L_{c2}, L_{r1}, L_{r2}, L_{p1}, L_{p2}\}$. L_{c1} and L_{c2} represent the large and the small cubic links (or cubic link modules) respectively; L_{r1} and L_{r2} represent the large and the small revolute links (or revolute link modules) respectively; and L_{p1} and L_{p2} represent the large and the small prismatic links (or prismatic link modules) respectively. Similarly, the joint set, J contains nine elements: $J = \{J_{r1}, J_{r2}, J_{r3}, J_{p1}, J_{p2}, J_{p3}, J_{f1}, J_{f2}, J_{f3}\}$, where J_{ri}, J_{pi}, and J_{fi} represent the revolute joint, the prismatic joint, and the fixed joint respectively. The subscript i indicates the size of a joint, i.e., large ($i = 1$), medium ($i = 2$), and small ($i = 3$) joints. Large, medium, and small joints are derived from the large connector, the adapter, and the small connector respectively. Note that, hereafter, the term *link* and *joint* will indicate the redefined ones if not specified. Based on the actual module designs, the redefined links are shown in Fig. 3.6. For our modeling purpose, each link is attached with a body-fixed frame (or the

module frame). The origin of this frame is located at the centroid of the cubic part of a link. The coordinate axes are perpendicular to the faces of modules. If the link is L_{ri} or L_{pi} $(i = 1, 2)$, for convenience, the z-axis of the link frame is defined to be along the direction of moving sockets.

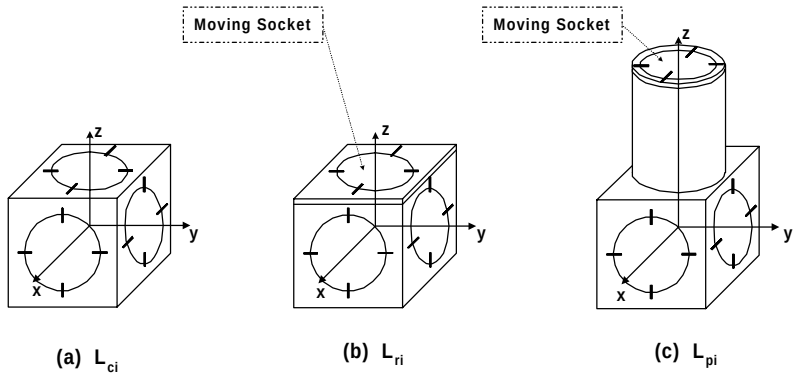


Figure 3.6: Redefined links

After this re-classification, a modular robot structure can be transformed into a common kinematic chain, and thus can be converted into a kinematic graph.

Example 3.7: Fig. 3.7(a) shows the assembly configuration of a modular robot. Its kinematic graph is a specialized graph, as shown in Fig 3.7(b). The eIM (Extended Incidence Matrix) of this graph is given by

$$\mathcal{M}(\tilde{G}) = \begin{bmatrix} 1 & 0 & 0 & 0 & L_{c1} \\ 1 & 1 & 0 & 0 & L_{r1} \\ 0 & 1 & 1 & 0 & L_{r1} \\ 0 & 0 & 1 & 1 & L_{p2} \\ 0 & 0 & 0 & 1 & L_{c2} \\ J_{f1} & J_{r1} & J_{r2} & J_{p3} & 0 \end{bmatrix}.$$

3.4 Assembly Incidence Matrix

A modular robot usually consists of modules of different types and sizes. Thus, it can be topologically described by a specialized graph, and then represented by an eIM. However, the eIM cannot fully represent the assembly configuration of a modular robot because it does not contain the assembly information such as the locations of the connected sockets

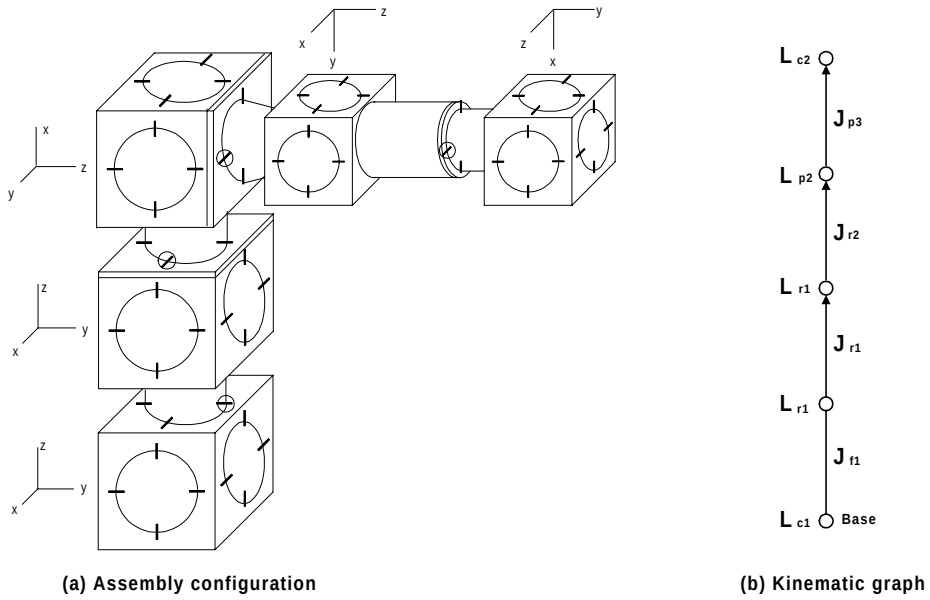


Figure 3.7: A modular robot assembly configuration

with respect to the modules and the relative assembly orientation of connected links. Hence, a modified eIM is to be proposed in order to represent the assembly configuration of a modular robot in a matrix form. Since each of the 1 entries in the eIM defines the connectivity of a link and a joint, a possible approach to modify the eIM is to replace the 1s with specially defined labels to indicate the assembly information between links and joints.

As mentioned in Chapter 2, a joint is allowed to attach a connecting socket of a link in four different orientations as shown in Fig. 3.8. These four symmetric orientations are marked by the numbers 1, 2, 3, and 4 corresponding to four different positions of the location holes. In other words, when the connecting socket and the position of the locating pin are given, the assembly of a link and a joint is uniquely determined. Therefore, the label to be defined should be capable of indicating any connecting socket and the position of the locating pin on a link. Based on the cubic or cube-like shape of links shown in Fig. 3.6, the label is defined by an ordered pair of directional parameters, termed a port vector $P = (p_1,p_2)$, where p_1 represents the normal direction of the connecting port on a link that the connector is engaged, and p_2 describes the position of the locating pin, i.e., the approaching orientation of the connected joint. The directional parameters p_1 and p_2 are all with respect to the link module frame so that $p_1, p_2 \in \{\pm x,\pm y,\pm z\}$, where $\pm x,\pm y$,

and $\pm z$ represent the positive and negative directions of x, y, and z axes of the link frame respectively and $p_1 \neq p_2$. By taking advantage of the inherent relations of the connecting sockets, positions of the location pin, and link frame, the defined labels can conveniently describe the assembly information. For instance, if the connecting socket is on the top surface of the module and the location pin occurs at position "1" as shown in Fig. 3.8, the corresponding port vector for this assembly pattern is given by $P = (+z, -y)$. Note that the representation and assignment of the labels are non-unique. The labels of the connecting sockets can also be represented by other symbols such as numbers [14].

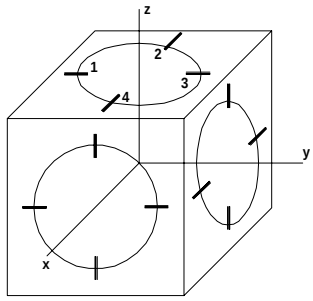


Figure 3.8: Relative orientations of a joint attaching to a link's connecting port

Definition 14 (Assembly Incidence Matrix [14, 17]) *Let $\tilde{G}$ be the specialized graph of a modular robot and $\mathcal{M}(\tilde{G})$ be its eIM. The assembly incidence matrix (AIM) of this robot $\mathcal{A}(\tilde{G})$ is defined by substituting each of 1s in $\mathcal{M}(\tilde{G})$ with the corresponding port vector $P = (p_1, p_2)$, where $p_1, p_2 \in \{\pm x, \pm y, \pm z\}$ and $p1 \neq p2$.*

Example 3.7: The AIM of the modular robot shown in Fig. 3.7(a) is given by

$$\mathcal{A}(\tilde{G}) = \begin{bmatrix} (+z,+y) & 0 & 0 & 0 & L_{c1}(Base) \\ (-z,+y) & (+z,+x) & 0 & 0 & L_{r1} \\ 0 & (-x,+y) & (+z,+y) & 0 & L_{r1} \\ 0 & 0 & (-z,+x) & (+z,+x) & L_{p2} \\ 0 & 0 & 0 & (-y,+z) & L_{c2} \\ J_{f1} & J_{r1} & J_{r2} & J_{p3} & 0 \end{bmatrix}.$$

Although a modular robot system can render many possible kinematic structures such as tree and closed loop types, we focus on the tree-structured robots that both serial and star-like robots belong to. Since the base module in a specific robot assembly is always known, we, for convenience, always arrange the rows in an AIM in the BFS order. On

the other hand, once such an AIM is given, it is very easy to derive the incidence matrix of its underlying graph by first deleting the last column and last row of the AIM and then setting all the nonzero entries to 1. It follows that both the accessibility and path matrices of a (rooted) tree-structured modular robot can be derived from the given AIM automatically. Note that the AIM can also be used to represent closed-loop modular robot configurations without modification [14].

3.5 Discussion

In this chapter, several basic issues related to the representation of modular robot assembly configurations have been reviewed and discussed. Among them, the most important concept is the *Assembly Incidence Matrix* (AIM) proposed by Chen [14]. The AIM can fully describe the assembly configuration of a modular robot in a compact matrix form. With such an AIM representation, the automatic generation of the kinematic, dynamic, and calibration models for a modular robot becomes more convenient.

Chapter 4

Modular Robot Kinematics

The kinematics of a modular robot describes the spatial relationship of the joints and links in the robot. This chapter focuses on two fundamental issues in modular robot kinematics: (1)*forward kinematics* - the determination of the end-effector (or end link) pose (position and orientation) as a function the input joint angles, and (2) *inverse kinematics* - the determination of the sets of joint angles corresponding to a given end-effector pose.

Previous researches on robot kinematics emphasized on serial type robots with fixed geometry. By employing the well developed kinematic modeling method, the Denavit-Hartenberg (D-H) parameterization method [32], we can formulate the kinematics of a conventional robot based on the specific robot geometry. A modular robot, however, can assume many possible configurations. Thus manual derivation of the kinematic models for all of the possible configurations on a case by case basis is obviously a tedious task. Based on the local frame representation of the Product-of-Exponentials (POE) formula - the local POE formula, a recursive forward kinematic algorithm and a numerical inverse kinematic algorithm are proposed in this chapter. These two algorithms can create the forward and inverse kinematic models from a given modular robot configuration automatically. Moreover, the algorithms are independent of the robot geometry and suitable for tree-structured modular robots.

This chapter is organized as follows. Section 4.1 briefly reviews the significant research results in robot kinematics. The Product-of-Exponentials (POE) formula as well as its geometric background is introduced in Section 4.2. In Section 4.3, a recursive forward

kinematic algorithm is proposed. The numerical inverse kinematic algorithm is then presented in Section 4.4.

4.1 Introduction

Robot kinematics is one of the fundamental issues in robotics. It studies the motion of robots for programming, control, and design purposes. Based on the geometric relationships of links and joints, several effective kinematic modeling methods, such as the D-H method [32], the screw coordinates method [37], the zero reference position method [48], and the POE formula [11] have been developed for conventional robots. Among them, the D-H representation and the POE formula are two significant milestones in robot kinematics.

The D-H method [32] uses a 4×4 homogeneous transformation matrix to describe the spatial relationship between adjacent links. The forward kinematics problem becomes finding an equivalent 4×4 homogeneous transformation matrix that relates the spatial displacement of the end-effector coordinate frame to the base reference coordinate frame. Based on particularly assigned link frames, the number of geometric parameters in the D-H representation of a robot is minimized such that only four geometric parameters, including the joint angle, are required to formulate the homogeneous transformation matrix between two adjacent links. The D-H method is algorithmically universal in deriving the kinematic equation of a robot, and is, therefore, well accepted. However, the D-H representation results in a set of rigidly defined coordinate frames in a robot so that arbitrary assignment of link coordinate frames is inconvenient. Also, the D-H parameters are configuration dependent. The same robot may have different sets of D-H parameters just because of the different initial configurations. For a modular robot, the reconfigurability calls for a configuration independent modeling method which is more flexible and adaptable.

Based on the screw theory, Brockett [11] shows that the kinematic equations for an open kinematic chain containing either revolute joints or prismatic joints can be expressed by a product of matrix exponentials (see Section 4.2 for more details). Because of its connection with Lie groups and Lie algebras, the POE formula provides a modern geometric

interpretation of the classical screw theory. By drawing upon well-established principles of the classical screw theory and the modern differential geometry, the POE formula not only has the intuitive simplicity and theoretical appeal, but can also be used as a computationally effective tool [11, 79, 90, 99, 105]. There are several advantages of using POE formula in robot kinematics modeling. The first advantage is that the POE formula provides a general and uniform representation of the forward kinematics of a spatial mechanism regardless of the types of joints. The second advantage is that the POE formula embodies all the advantages of the screw coordinates method, and allows a global description of the rigid motion which does not suffer from singularities due to the use of local coordinates. Such singularities are inevitable when one chooses to represent the rotation via Euler angles. The third advantage is that the POE formula is basically a zero reference position method so that it also embodies all the advantages of the zero reference position method. The description of the robot is in terms of the joint axis (twists) directions and locations in the zero reference position. Hence, the local coordinate frames in POE representation can be arbitrarily defined. Due to these advantages, the POE formula proved to be a more convenient and powerful modeling method for a modular reconfigurable robot system in which the kinematics, dynamics, and calibration algorithms are to be independent of robot configurations.

The derivation of the forward kinematics for a conventional robot is quite straightforward once a kinematic modeling method is chosen. For a modular robot, the forward kinematics should be formulated automatically once a specific robot configuration (an AIM) is given. Researches in CMU [66, 115] and University of Toronto [7] have proposed techniques, based on the D-H representation, to generate modular robot forward kinematics automatically. The method of [66, 115] uses 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 notation. According to a given connecting sequence of modules, the D-H parameters of the entire robot can be obtained automatically by employing a conversion algorithm from the modular frames to D-H frames. However, only revolute joints are considered in the kinematic parameter conversion scheme. The work of [7] generalized this modeling technique to include both

revolute joints and prismatic joints. Also, the multiple attachment of joints on one link module is considered by defining input/output frames on every link.

Based on the POE formula, another kinematic modeling approach is proposed by Chen and Yang[15], which can generate the forward kinematics of a tree-structured modular robot from a given AIM automatically. In this technique, only one set of local coordinate frames, termed *module frames* is required to describe the motion of a modular robot. Such a local frame is naturally assigned to each of the modules as shown in Fig. 3.6. A modular robot assembly is then represented by its intrinsic properties, i.e., the locations of the joint axes and the types of the joints, which can be conveniently derived from an AIM by means of a *port conversion* scheme. The dyad kinematics that relates to the motion of two consecutive modules under a joint displacement is also proposed. Using dyad kinematics along with a tree-traversing algorithm to search for a given AIM, one can derive the forward kinematics automatically.

The inverse kinematics problem consists of the determination of joint variables corresponding to a given end-effector position and orientation. The solution to this problem is of fundamental importance in order to transform the motion specifications assigned to the end-effector in the task space into the corresponding joint space motions that allow the execution of the desired motion.

There are two main approaches, namely, the analytical method and the numerical method, which have been well addressed in the inverse kinematics literatures. The main advantages of the analytical approach over the numerical one are the derivation of computationally efficient closed-form solutions, where each joint variable is expressed in terms of other known quantities, and all the solutions can be found simultaneously [30, 67, 108, 107]. However, the existence of closed form inverse kinematics solutions depends on the kinematic structure of the robot. For example, a sufficient condition for a 6-DOF serial type robot to have a closed form solution is that the three adjacent joint axes intersect together. Therefore, when a general-purpose program for solving the inverse kinematics problem is needed, the numerical method is the preferred approach for solution. Many numerical algorithms have been developed so far. Most of them are based on multi-dimensional Newton-Raphson iteration scheme or similar techniques such as the modified predictor-corrector algorithm,

the conjugate gradient algorithm, and the separation of variable method to provide a solution [33, 40, 49, 78, 83, 85, 140]. In [4, 45, 65], the inverse kinematic problem is modified as an optimization problem, and numerical optimization algorithms are thus employed to search for the solutions. Most of these inverse kinematics models are based on the D-H notation, and only fixed-configuration, serial type robots are considered.

Note that, in the past two decades, there has also been an increasing interest in the symbolic computation of the inverse kinematics of a serial, articulated robot, especially, the $6R$ robot. Most of these studies have been focused on the derivation of high degree polynomials in the tangent of the half-angle of joint variables, and implementation of efficient and robust algorithms to find all of the roots of the polynomial [3, 38, 74, 75, 76, 82, 110, 127]. Although this approach is general, only serial robots with revolute joints are considered, and at present it is still slow for on-line computations.

For a modular robot system, one can build numerous robots with distinct geometries out of an inventory of robot modules [14]. Therefore, it is impossible and impractical to derive the kinematics model of every modular robot configuration case by case and store the models as library packages in the computer. Khosla *et al* [66], based on the D-H notation, proposed a numerical inverse kinematic algorithm for serial type modular robots. In this chapter, based on the local POE formula, we present a systematic modeling scheme which can generate the inverse kinematic model of a modular robot with tree-structured geometry automatically [22]. In this scheme, we use a *path* matrix of the underlying robot geometry to assist the construction of the inverse kinematics model. Some variations have also been made on the inverse kinematic model for dealing with the pure position, pure orientation, and hybrid inverse kinematics problems. According to the differential form of the resulting inverse kinematic model, a numerical approach using Newton-Raphson iteration method is employed. Note that the main objective of this scheme, like the work of Featherstone [40], is to increase the generality of the inverse kinematic algorithm, which is very important for modular robot systems.

4.2 Geometric Background and the POE Formula

In this section, the Product-of-Exponentials (POE) formula as well as the geometric background is reviewed. For more details, please refer to [11, 79, 90, 100]

4.2.1 Geometric Background

In robot kinematics, it is sufficient to think of $SE(3)$, the *Special Euclidean Group* of rigid body motions, as consisting of matrices of the form

$$\begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}, \tag{4.1}$$

where $R \in SO(3)$ is interpreted as a rigid body rotation and $p \in \Re^{3\times 1}$ as a rigid body translation. Here, $SO(3)$, the *Special Orthogonal Group*, denotes the group of 3×3 rotation matrices. Elements of $SE(3)$ can also be denoted by the ordered pair (p, R), with group multiplication understood to be $(p_1, R_1).(p_2, R_2) = (R_1 p_2 + p_1, R_1 R_2)$. $SE(3)$ is a Lie group of dimension six.

The *Lie algebra* of $SE(3)$, denoted $se(3)$, consists of matrices of the form

$$\begin{bmatrix} \hat{w} & v \\ 0 & 0 \end{bmatrix}, \tag{4.2}$$

where

$$\hat{w} = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}. \tag{4.3}$$

The set of 3×3 real skew-symmetric matrices, $\hat{w}$, forms the Lie algebra of $SO(3)$, denoted by $so(3)$. Note that an element of $\hat{w} \in so(3)$ can also be regarded as a vector $w \in \Re^{3\times 1}$. Since in most cases it will be clear from the context which representation is implied, an element of $so(3)$, $\hat{w}$, will also be simply denoted by w in such cases. An element of $se(3)$ will thus admit a 6×1 vector presentation: $(v, w) \in \Re^{6\times 1}$, termed a twist. The twist denotes the line coordinate of the *screw* axis of a general rigid body motion (rotation and translation) in which w is the unit directional vector of the axis and v is the position of the axis relative to the origin.

On matrix Lie algebras, the *Lie bracket* is given by the matrix commutator: if A and B are elements of a matrix Lie algebra, then $[A, B] = AB - BA$. In particular, on $so(3)$ the

Lie bracket of two elements corresponds to their vector product: $[w_1, w_2] = w_1 \times w_2$. On $se(3)$, the Lie bracket of two elements (v_1, w_1) and (v_2, w_2) is given by

$$[(v_1, w_1), (v_2, w_2)] = (w_1 \times v_2 - w_2 \times v_1, w_1 \times w_2). \tag{4.4}$$

An element of a Lie group can also be identified with a linear mapping between its Lie algebra via the *adjoint representation*. Suppose G is a matrix Lie group with Lie algebra g. For every $X \in G$ the adjoint map $Ad_X : g \to g$ is defined by $Ad_X(x) = XxX^{-1}$ for $x \in g$. If $X = (p, R)$ is an element of $SE(3)$, then its adjoint map acting on an element $x = (v, w)$ of $se(3)$ is given by

$$Ad_X(x) = (p \times Rw + Rv, Rw), \tag{4.5}$$

which also admits the 6×6 matrix representation

$$Ad_X(x) = \begin{bmatrix} R & \hat{p}R \\ 0 & R \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{4.6}$$

It is easily verified that $Ad_X^{-1} = Ad_{X^{-1}}$ and $Ad_X Ad_Y = Ad_{XY}$ for any $X, Y \in SE(3)$.

Elements of a Lie algebra can also be identified with a linear mapping between its Lie algebra via the Lie bracket. Given an element $x \in g$, its adjoint representation is the linear map $ad_x : g \to g$ defined by $ad_x(y) = [x, y]$. If $x = (v_1, w_1)$ and $y = (v_2, w_2)$ are elements of $se(3)$, then

$$ad_x y = (w_1 \times v_2 - w_2 \times v_1, w_1 \times w_2), \tag{4.7}$$

which also admits the matrix representation

$$ad_x y = \begin{bmatrix} \hat{w}_1 & \hat{v}_1 \\ 0 & \hat{w}_1 \end{bmatrix} \begin{bmatrix} v_2 \\ w_2 \end{bmatrix}. \tag{4.8}$$

An important connection between a Lie group, $SE(3)$, and its Lie algebra, $se(3)$, is the exponential mapping, defined on each Lie algebra. Let $\hat{s} \in se(3)$ ($s = (v, w)$), and $\|w\|^2 = w_x^2 + w_y^2 + w_z^2$, then

$$e^{\hat{s}} = \begin{bmatrix} e^{\hat{w}} & Av \\ 0 & 1 \end{bmatrix} \in SE(3), \tag{4.9}$$

where

$$e^{\hat{w}} = I + \frac{\sin\|w\|}{\|w\|}\hat{w} + \frac{1 - \cos\|w\|}{\|w\|^2}\hat{w}^2 \tag{4.10}$$

$$A = I + \frac{1 - \cos\|w\|}{\|w\|^2}\hat{w} + \frac{\|w\| - \sin\|w\|}{\|w\|^3}\hat{w}^2. \tag{4.11}$$

The matrix logarithm also establishes a connection between a Lie group, $SE(3)$, and its Lie algebra,$se(3)$ [100]. Let $R \in SO(3)$ such that trace$(R) \neq -1$, $1 + 2\cos\phi = \text{trace}(R)$, and $\|\phi\| < \pi$, then

$$\log \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \hat{w} & A^* p \\ 0 & 0 \end{bmatrix} \in se(3), \tag{4.12}$$

where

$$\hat{w} = \log R = \frac{\phi}{2\sin\phi}(R - R^T) \tag{4.13}$$

$$A^* = I - \frac{1}{2}\hat{w} + \frac{2\sin\|w\| - \|w\|(1 + \cos\|w\|)}{2\|w\|^2 \sin\|w\|}\hat{w}^2. \tag{4.14}$$

If ϕ is very small, $\hat{w} \approx (R - R^T)/2$. If the trace$(R) = -1$, it can be shown that $\log R = (2k+1)\pi\hat{v}$, where k is any integer and v is the unit eigenvector of $\log R$ corresponding to the eigenvalue 1.

An actual example of Lie algebra in robot kinematics is the generalized velocities of rigid bodies [90, 100]. There are two ways in which the tangent vector $\dot{X}(t)$ of a curve $X(t) = (p(t), R(t)) \in SE(3)$ can be identified with an element of $se(3)$. If $X(t)$ describes the motion of a rigid body relative to a spatial reference frame, then both $\dot{X}X^{-1}$=$(\dot{p} - \dot{R}R^{-1}p, \dot{R}R^{-1})$ and $X^{-1}\dot{X} = (R^{-1}\dot{p}, R^{-1}\dot{R})$ are elements of $se(3)$. $X^{-1}\dot{X}$ is referred to as the *body velocity* representation of $\dot{X}$, since $R^{-1}\dot{R}$ and $R^{-1}\dot{p}$ are the angular velocity and translational velocities of the rigid body relative to its body-fixed frame respectively. Similarly, $\dot{X}X^{-1}$ is referred to as the *spatial velocity* representation of $\dot{X}$. One subtle and important difference in the interpretation of the spatial velocity is that $R^{-1}\dot{R}$ is indeed the angular velocity of the rigid body relatively to the spatial frame, while the translational velocity relative to the spatial frame is not $\dot{p} - \dot{R}R^{-1}p$, but simply $\dot{p}$. If $X(t)$ undergoes a coordinate transformation of the form $X(t) \mapsto X(t)T$, where $T \in SE(3)$ is constant, then its new body velocity representation is $T^{-1}X^{-1}\dot{X}T = Ad_T(X^{-1}\dot{X})$.

4.2.2 The POE Formula

We now briefly review the Product-of-Exponentials (POE) formula for open chain robots in which the links form a single serial chain and each pair of links is connected either by a revolute joint or a prismatic joint [11, 90, 100]. For convenience, we temporarily

number the joints from 1 to n, starting at the base, and number the links such that joint i connects links $i-1$ and i. Link 0 is taken to be the base and link n is taken to be the end link. If a right-handed reference frame is fixed at each link of the chain, then the element of $SE(3)$ describing the position and orientation of frame i relative to that of frame $i-1$ is $T_{i-1,i}(q_i)= T_{i-1,i}(0)e^{\hat{s}_{i,i}q_i}$, where $T_{i,i-1}(0) \in SE(3)$ is the initial position and orientation of frame i relative to frame $i-1$, $\hat{s}_{i,i}$ or simply $\hat{s}_i \in se(3)$ represents the twist of joint i's axis expressed in frame i, and $q_i \in \Re R$ is the joint i's displacement. The frame fixed at the end-effector or the end-link is then related to that of the base by the product

$$\begin{aligned} T_{0n}(q_1, q_2, \ldots, q_n) &= T_{0,1}(q_1)T_{1,2}(q_2)\ldots T_{n-1,n}(q_n) \\ &= T_{0,1}(0)e^{\hat{s}_1 q_1}T_{1,2}(0)e^{\hat{s}_2 q_2}\ldots T_{n-1,n}(0)e^{\hat{s}_n q_n}. \end{aligned} \tag{4.15}$$

If the twist of joint i's axis is expressed in frame $i-1$, denoted by $\hat{s}_{i-1,i}$, Equation (4.15) can be rewritten as

$$T_{0n}(q_1, q_2, \ldots, q_n) = e^{\hat{s}_{0,1}q_1}T_{0,1}(0)e^{\hat{s}_{1,2}q_2}T_{1,2}(0)\ldots e^{\hat{s}_{n-1,n}q_n}T_{n-1,n}(0), \tag{4.16}$$

where $\hat{s}_{i-1,i} = T_{i-1,i}(0)\hat{s}_i T_{i-1,i}^{-1}(0) = Ad_{T_{i-1,i}(0)}s_i$. If all the twists are expressed in the base frame 0, Equation (4.15) can also be written as

$$T_{0n}(q_1, q_2, \ldots, q_n) = e^{\hat{s}_{0,1}q_1}e^{\hat{s}_{0,2}q_2}\ldots e^{\hat{s}_{0,n}q_n}T_{0,n}(0), \tag{4.17}$$

where $\hat{s}_{0,i} = T_{0,i}(0)\hat{s}_i T_{0,i}^{-1}(0) = Ad_{T_{0,i}(0)}\hat{s}_i$ in which $T_{0,i}(0) = T_{0,1}(0)T_{1,2}(0)\ldots T_{i-1,i}(0)$. $T_{0,n}(0) = \prod_{i=1}^{n} T_{i-1,i}(0)$ represents the initial position and orientation of the end-effector frame relative to the base frame.

If all of the twists are expressed in the end-effector (end link) frame n, Equation (4.15) can also be rewritten as

$$T_{0n}(q_1, q_2, \ldots, q_n) = T_{0,n}(0)e^{\hat{s}_{n,1}q_1}e^{\hat{s}_{n,2}q_2}\ldots e^{\hat{s}_{n,n}q_n}, \tag{4.18}$$

where $\hat{s}_{n,i} = T_{i,n}^{-1}(0)\hat{s}_i T_{i,n}(0) = Ad_{T_{i,n}^{-1}(0)}\hat{s}_i$ and $T_{i,n}(0) = T_{i,i+1}(0)T_{i+1,i+2}(0)\ldots T_{n-1,n}(0)$.

Equations (4.15)-(4.18) are all called the Product-of-Exponentials (POE) formula, in which Equation (4.15) and (4.16) are called the local frame representation of POE formula, i.e., the local POE formula. The POE formula provides a clear geometric representation of the kinematics of a robot and can simplify the kinematic analysis of the mechanism [104].

4.3 Forward Kinematics

In this section, we focus on the forward kinematics of tree-structured modular robots which most of the industrial robots belong to. A tree-structured kinematic chain generally consists of several branches (from the base to each of the pendent links). The forward kinematics of an n-DOF tree-structured modular robot can thus be described as: given a set of joint angles $q = \{q_1, q_2, \ldots, q_n\}$, we wish to determine the poses of all the end-effector frames relatively to the unique base frame. Although different branches may have several common links and joints, in formulating the forward kinematics, each branch can still be considered as a serial chain. Since each joint connects two and only two consecutive links, it is reasonable to first consider the description of kinematic relationship between two consecutive links and then obtain the overall description of robot kinematics in a recursive fashion. Therefore, the local frame representation of POE formula is more suitable for our purpose. Note that in the following context, we consider modular robots without end-effectors, so that the desired poses are referred to as the poses of end link frames.

4.3.1 Dyad Kinematics

As a first step, a systematic and general method is to be derived in order to define the relative position and orientation of two consecutive links. The problem is to determine two frames attached to the two links and compute the coordinate transformations between them. In general, these two frames can be arbitrarily chosen as long as they are attached to the link they are referring to. Since there exists a local frame (module frame) on each of the links, it is unnecessary and inconvenient to define new frames on the links. For convenience, we adopt these naturally defined module frames as the local frames throughout this thesis.

A dyad is referred to as a pair of consecutive links in a kinematic chain. Based on the local POE formula, i.e., Equation (4.15), the dyad kinematics which relates the motion of two connected modules under a joint displacement is defined as follows:

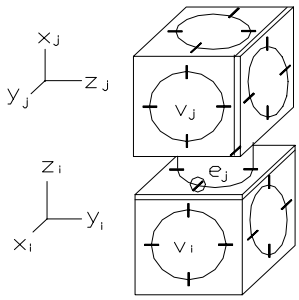


Figure 4.1: Two connected modules (a dyad)

Let v_i and v_j be two adjacent links connected by a joint e_j, as shown in Fig. 4.1 and also consider joint e_j and link v_j as *link assembly* j. Denote the module coordinate frames on link v_i as frame i and link v_j as frame j. Then the relative pose (position and orientation) of frame j with respect to frame i, under a joint displacement, q_j, can be described by a 4×4 homogeneous matrix, an element of $SE(3)$, such that

$$T_{ij}(q_j) = T_{ij}(0)e^{\hat{s}_j q_j}, \tag{4.19}$$

where $\hat{s}_j \in se(3)$ is the twist of joint e_j expressed in frame j, $T_{ij}(q_j)$ and $T_{ij}(0) \in SE(3)$. Note that in the following context, the pose of a coordinate frame is referred to as the 4×4 homogeneous matrix including both the orientation and position of the frame with respect to a reference frame. In Equation (4.19), $T_{ij}(0)$ is the initial pose of frame j relative to frame i, and

$$T_{ij}(0) = \begin{bmatrix} R_{ij}(0) & d_{ij}(0) \\ 0 & 1 \end{bmatrix}, \tag{4.20}$$

where $R_{ij}(0) \in SO(3)$ and $d_{ij}(0) \in \Re^{3\times 1}$ are the initial orientation and position of link frame j relative to frame i respectively. The twist of link assembly j is written as

$$\hat{s}_j = \begin{bmatrix} \hat{w}_j & v_j \\ 0 & 0 \end{bmatrix}, \tag{4.21}$$

where $\hat{w}_j$ is a skew-symmetric matrix related to $w_j (= (w_{jx}, w_{jy}, w_{jz})^T)$, the unit directional vector of the joint axis expressed in frame j, and $v_j (= (v_{jx}, v_{jy}, v_{jz})^T)$ is the position vector of the joint axis relative to frame j. $\hat{w}_j$ is given by

$$\hat{w}_j = \begin{bmatrix} 0 & -w_{jz} & w_{jy} \\ w_{jz} & 0 & -w_{jx} \\ -w_{jy} & w_{jx} & 0 \end{bmatrix}. \tag{4.22}$$

The twist, $\hat{s}_j$, can also be represented by a 6-dimensional vector through a mapping: $\hat{s}_j \mapsto (v_j, w_j) \in \Re^{6\times 1}$, termed twist coordinates. Based on the module design, the joint

j's axis always passes through frame j's origin. For revolute joints, $s_j = (0, w_j)$, where w_j is the unit directional vector of the joint axis expressed in frame j. For prismatic joints, $s_j = (v_j, 0)$, where v_j is the unit directional vector of the joint axis relative to frame j. For fixed joints, $s_j = (0, 0)$. Therefore, Equation (4.9) can also be simplified as:

$$e^{\hat{s}_j q_j} = \begin{bmatrix} e^{\hat{w}_j q_j} & v_j q_j \\ 0 & 1 \end{bmatrix}, \tag{4.23}$$

where q_j is joint j's displacement and

$$e^{\hat{w}_j q_j} = I + \hat{w}_j \sin q_j + \hat{w}_j{}^2 (1 - \cos q_j). \tag{4.24}$$

Since a specific modular robot is given in an AIM (Assembly Incidence Matrix) format as mentioned in the previous chapter, the initial orientation $R_{ij}(0)$, initial position $d_{ij}(0)$, and the twist coordinate s_j of each dyad are to be determined from the given AIM in order to generate the dyad kinematics automatically. In an AIM, as shown in Equation (4.25), five entries are related to the dyad of links $\mathrm{v_i}$ and $\mathrm{v_j}$, i.e., $(a_{ik}, a_{in}, a_{jk}, a_{jn}, a_{n+1,k})$, termed a dyad vector. $a_{n+1,k}$ represents the type of joint e_k; a_{in} and a_{jn} indicate the type of link $\mathrm{v_i}$ and $\mathrm{v_j}$ respectively; $a_{ik} = \{p_{ik}, q_{ik}\}$ and $a_{jk} = \{p_{jk}, q_{jk}\}$ represent that the a_{ik}-port of $\mathrm{v_i}$ is connected with a_{jk}-port of $\mathrm{v_j}$ by a joint $\mathrm{e_k}$.

$$\mathcal{A}(\widetilde{G}) = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & a_{ik} & \cdots & a_{in} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & a_{jk} & \cdots & a_{jn} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & a_{n+1,k} & \cdots & 0 \end{bmatrix} \tag{4.25}$$

- **Determination of Initial Orientation** $R_{ij}(0)$

  The columns of $R_{ij}(0)$ represent the initial orientation of the axes of frame j relative to frame i. Furthermore, the first, second, and third column of $R_{ij}(0)$, denoted by $R_{ij}^m(0)(m = 1, 2, 3)$, represent the unit directional vector of x, y, and z-axis of frame j relative to frame i initially. Let $P_{ik} = a_{ik} = (p_{ik}, q_{ik})$ and $P_{jk} = a_{jk} = (p_{jk}, q_{jk})$ be the port vectors of the two connected modules i and j respectively. Since these two connecting sockets are facing each other, the direction of p_{jk} of frame j is opposite to the direction of the p_{ik} of frame i. According to this relationship, the direction vector of one axis of frame j (related to p_{jk}) with respect to frame i can be determined,

which is opposite to the direction of p_{ik} and then put in the corresponding column of $R_{ij}(0)$ according to p_{jk}. For instance, in Fig 4.2, if $P_{ik} = (p_{ik},\, q_{ik}) = (+z_i, +x_i)$ and $P_{jk} = (p_{jk},\, q_{jk}) = (-x_j, +y_j)$, then the direction of x-axis of frame j, x_j, is now in the z-direction of frame i, $+z_i$, so that the vector (0, 0, 1) should be put into the first column of $R_{ij}(0)$.

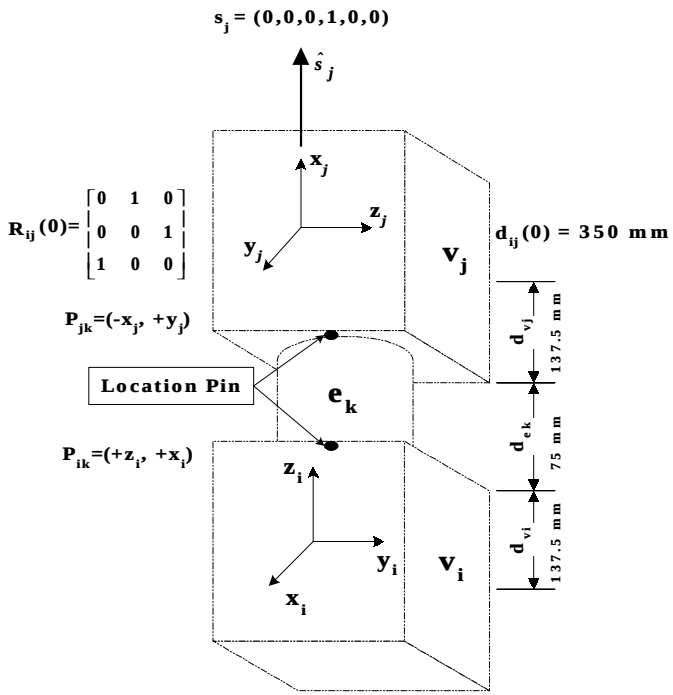


Figure 4.2: Determination of the initial pose of a dyad

In the port vectors, q_{ik} and q_{jk} represent the attaching orientations of joint e_k on link v_i and v_j respectively, which are corresponding to the positions of the location pins. They are in the same physical direction according to our definition. Hence, the direction of another axis of frame j (related to q_{jk}) with respect to frame i can also be determined. It is in the direction of q_{ik} and then put into the corresponding column of $R_{ij}(0)$ according to q_{jk}. Following the previous example (Fig. 4.2), since the y direction of frame j, y_j, is in the x direction of frame i, the unit vector (1, 0, 0) is placed in the second column of $R_{ij}(0)$.

According to the port vectors of two connected links, we can get two column vectors of the initial orientation matrix $R_{ij}(0)$. Since $R_{ij} \in SO(3)$ is an orthogonal matrix, the remaining column vector can be calculated by the cross product of the two

known column vectors using one of the following equations:

$$R^1_{ij}(0) = R^2_{ij}(0) \times R^3_{ij}(0); \quad (4.26)$$

$$R^2_{ij}(0) = R^3_{ij}(0) \times R^1_{ij}(0); \quad (4.27)$$

$$R^3_{ij}(0) = R^1_{ij}(0) \times R^2_{ij}(0); \quad (4.28)$$

where the superscript k ($k = 1, 2, 3$) represents the column order.

- **Determination of Initial Displacement** $d_{ij}(0)$

  The initial displacement vector represents the position of origin of frame j relative to frame i initially. $d_{ij}(0)= (d^x_{ij}, d^y_{ij}, d^z_{ij}) \in \Re^{3\times 1}$, where d^x_{ij}, d^y_{ij}, and d^z_{ij} represent the x, y, and z coordinates of frame j's origin relative to frame i respectively. Because all the connecting sockets of a module are centered on the faces of the cube and the module frame is assigned to the centroid of the cube, the origin of frame j is always in one of frame i's axes, i.e., along the direction of p_{ik}. Therefore, two elements in this vector must be zero. According to p_{ik}, we can first determine which element in $d_{ij}(0)$ is the non-zero entry, and also the entry's sign. The absolute value of the non-zero entry is equal to the distance between the origin of frame i and the origin of frame j, which can be calculated by

  $$||d_{ij}(0)|| = d_{\mathrm{vi}} + d_{\mathrm{vj}} + d_{\mathrm{ek}} \quad (4.29)$$

  where d_{vi} (or d_{vj}) is the distance from the origin of link frame i (or j) to the connecting surface of a_{ik}-port (or a_{jk}-port), and d_{ek} is the length of joint (connector) $\mathrm{e_k}$. All these dimensions can be derived from the module database according to a given dyad vector.

- **Determination of** s_j

  Here, s_j is the twist coordinate of joint $\mathrm{e_k}$, which is expressed in frame j. Since $\mathrm{v_i}$ is the ancestor of $\mathrm{v_j}$, the twist $\hat{s_j}$ should, according to our assembly rules, point to connecting socket of $\mathrm{v_j}$. As a result, the direction of $\hat{s_j}$ is opposite to the direction of p_{jk}. Hence, s_j can be determined by the following procedure:

  1. If $\mathrm{e_k}$ is a revolute joint, then $s_j = (0, w_j)^T$, where w_j is a unit directional vector in frame j representing $(-p_{jk})$.

2. If $\mathrm{e_k}$ is a prismatic joint, then $s_j = (\upsilon_j, 0)^T$, where υ_j is a unit directional vector in frame j representing $(-p_{jk})$.
3. If $\mathrm{e_k}$ is a fixed joint, then $s_j = (0, 0)^T$.

Example 4.1: As shown in Fig. 4.2, link $\mathrm{v_i}$ of type L_{r1} and link $\mathrm{v_j}$ of type L_{r1} are connected through joint $\mathrm{e_k}$ of type J_{r1}. Link $\mathrm{v_i}$ is the ancestor of Link $\mathrm{v_j}$. The connected port vectors of $\mathrm{v_i}$ and $\mathrm{v_j}$ are $P_{ik} = (+z_i, +y_i)$ and $P_{jk} = (-x_j, +y_j)$ respectively. Based on the algorithms mentioned above, we can obtain:
$R_{ij}(0) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$, $d_{ij}(0) = (0, 0, 350mm)$, and $s_j = (0, 0, 0, 1, 0, 0)^T$. Then the forward kinematics transformation from frame i to j can be calculated by Equation (4.19).

4.3.2 Forward Kinematics for a Tree-structured Modular Robot

The forward kinematics of a tree-structured robot is to determine the kinematic transformations from the base to each of the pendant links (or the end-effectors). Hence, finding out the connecting orders of links from a given AIM is essential. There are two tree-traversing algorithms usually used in graphs, the Breadth-First-Search (BFS) algorithm and the Depth-First-Search (DFS) algorithm. They can both be employed to determine the connecting orders of links as well as the corresponding dyad vectors from a given AIM.

An recursive forward kinematics algorithm for a tree-structured modular robot is proposed by Chen and Yang [15] by using the BFS algorithm to traverse a given AIM and applying Equation (4.19) to every dyad. The input of this algorithm is a given AIM and a set of joint angles; the output is the poses of all the end links with respect to the base frame. Alternatively, the forward kinematic equation can be derived by means of the *path* matrix. Since the rows in the AIM are arranged in BFS order, the *path* matrix can be derived from the given AIM as mentioned in Chapter 3. Once the *path matrix* is obtained, the dyad vectors can be derived from the AIM accordingly.

Now we consider a tree-structured modular robot consisting of $n+1$ modules and having m branches. The path matrix of this robot therefore has m rows and $n+1$ columns. Suppose that row k $(k = 1, 2, \ldots, m)$ has $n[k] + 1$ nonzero entries, sequentially 0_k(or

simplify to be 0 because of the unique base), $1_k, \ldots, n[k]_k$ from the left to right. Here i_k is the index of the ith non-zero entry in row k, and $i = 0, 1, \ldots, n[k]$. The forward kinematics of branch k thus can be given by:

$$T_{0,n[k]_k} = T_{0,1_k}(q_{1_k})T_{1_k,2_k}(q_{2_k}) \ldots T_{(n[k]-1)_k,n[k]_k}(q_{n[k]_k}) = \prod_{i=1}^{n[k]} (T_{(i-1)_k,i_k}(0)e^{\hat{s}_{i_k} q_{i_k}}) \tag{4.30}$$

The forward kinematic equation for the tree-structured modular robot with m branches is

$$T(q_1, q_2, \ldots, q_n) = \begin{bmatrix} T_{0,n[1]_1} \\ T_{0,n[2]_2} \\ \ldots \\ T_{0,n[m]_m} \end{bmatrix} = \begin{bmatrix} \prod_{i=1}^{n[1]} (T_{(i-1)_1,i_1}(0)e^{\hat{s}_{i_1} q_{i_1}}) \\ \prod_{i=1}^{n[2]} (T_{(i-1)_2,i_2}(0)e^{\hat{s}_{i_2} q_{i_2}}) \\ \ldots \\ \prod_{i=1}^{n[m]} (T_{(i-1)_m,i_m}(0)e^{\hat{s}_{i_m} q_{i_m}}) \end{bmatrix} \tag{4.31}$$

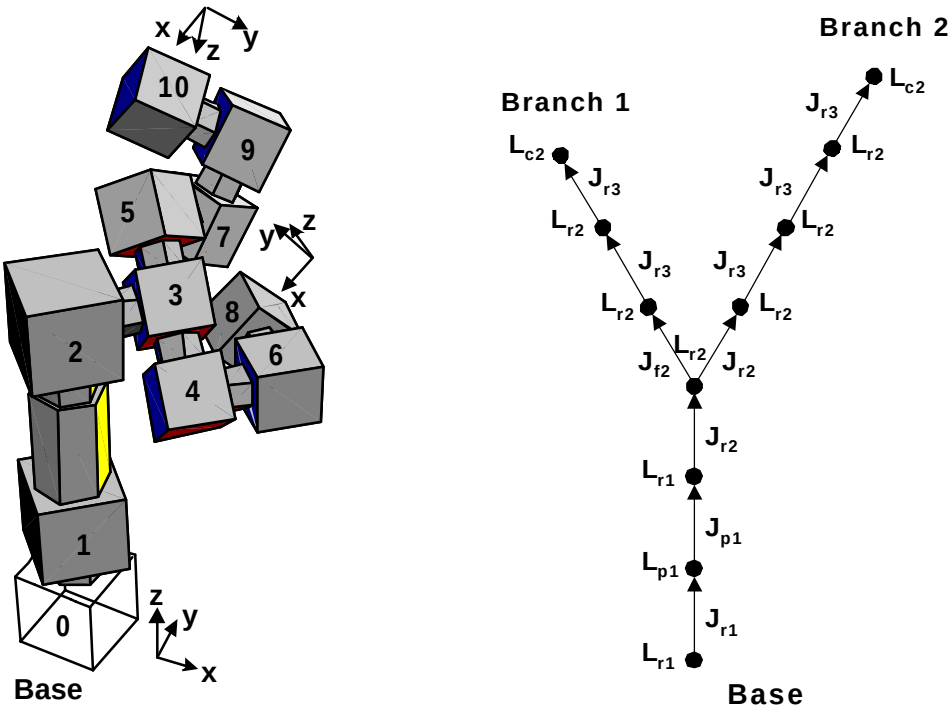


Figure 4.3: A tree-structured modular robot configuration (9-DOF)

Example 4.2 The AIM of a tree-structured modular robot (as shown in Fig. 4.3) is given by

$$\mathcal{A}(\tilde{G}) = \begin{bmatrix} (z,x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & L_{r1} \\ (-z,y) & (z,x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & L_{p1} \\ 0 & (-x,y) & (z,x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & L_{r1} \\ 0 & 0 & (x,y) & (-z,-y) & (z,y) & 0 & 0 & 0 & 0 & 0 & L_{r2} \\ 0 & 0 & 0 & (-x,y) & 0 & (z,y) & 0 & 0 & 0 & 0 & L_{r2} \\ 0 & 0 & 0 & 0 & (-y,-x) & 0 & (z,x) & 0 & 0 & 0 & L_{r2} \\ 0 & 0 & 0 & 0 & 0 & (y,z) & 0 & (z,x) & 0 & 0 & L_{r2} \\ 0 & 0 & 0 & 0 & 0 & 0 & (-x,y) & 0 & (z,y) & 0 & L_{r2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (-z,x) & 0 & 0 & L_{c2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (x,z) & (z,y) & L_{r2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (y,z) & L_{c2} \\ J_{r1} & J_{p1} & J_{r2} & J_{f2} & J_{r2} & J_{r3} & J_{r3} & J_{r3} & J_{r3} & J_{r3} & 0 \end{bmatrix}.$$

Using the algorithms proposed in the previous chapter, the path matrix of $\mathcal{A}(\tilde{G})$ can be derived:

$$\mathcal{P}(\vec{G}) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

According to the path matrix, we know that this robot has two branches. The first branch consists of links 0, 1, 2, 3, 4, 6, and 8; the second branch consists of links 0, 1, 2, 3, 5, 7, 9, and 10. Note that the sequential number of a link is the same as the row number of that link in the AIM. Using our forward kinematics algorithm, the kinematic transformations from the base (link 0) to link 8 and link 10 can be given by:

$$T(q_1, q_2, \ldots, q_{10}) = \begin{bmatrix} T_{0,8} \\ T_{0,10} \end{bmatrix}$$
$$= \begin{bmatrix} T_{0,1}(0)e^{\hat{s}_1 q_1}T_{1,2}(0)e^{\hat{s}_2 q_2}T_{2,3}(0)e^{\hat{s}_3 q_3}T_{3,4}(0)e^{\hat{s}_4 q_4}T_{4,6}(0)e^{\hat{s}_6 q_6}T_{6,8}(0)e^{\hat{s}_8 q_8} \\ T_{0,1}(0)e^{\hat{s}_1 q_1}T_{1,2}(0)e^{\hat{s}_2 q_2}T_{2,3}(0)e^{\hat{s}_3 q_3}T_{3,5}(0)e^{\hat{s}_5 q_5}T_{5,7}(0)e^{\hat{s}_7 q_7}T_{7,9}(0)e^{\hat{s}_9 q_9}T_{9,10}(0)e^{\hat{s}_{10} q_{10}} \end{bmatrix}.$$

Fig. 4.3 shows the robot configuration with joint angles $q = \{\frac{\pi}{4}, 100(mm), \frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4}\}$.

4.4 Inverse Kinematics

The purpose of an inverse kinematics algorithm is to determine the joint angles that cause a robot to reach a desired pose of the end-effector (the end link). Because a modular robot has unfixed assembly configurations and the number of DOFs in the robot varies, it is usually impossible to find closed-form inverse kinematics solution for a modular robot. Hence, a numerical approach is employed. In general, a numerical inverse kinematics model can be derived through linearizing the forward kinematic equation. Based on the

local POE formula and the definition of logarithm on $SE(3)$ and $SO(3)$, a compact and computational effective inverse kinematics model for a general tree-structured robot is derived by Chen and Yang [22]. This model can be easily modified to solve the pure position, pure orientation, and hybrid inverse kinematics problems. This section starts from formulating the inverse kinematics model of a single branch of a tree-structured robot. The obtained formula is suitable for serial and star-like robots which are the commonly used robot geometries. Then the inverse kinematics model as well as a closed-loop iterative algorithm is proposed for a complete tree-structured robot.

4.4.1 Differential Kinematics Model for a Single Branch

According to Equation (4.30), the linearized kinematic equation for branch k can be given by

$$dT_{0,n[k]_k} = \sum_{i=1}^{n[k]} \frac{\partial T_{0,n[k]_k}}{\partial q_{i_k}} dq_{i_k} \tag{4.32}$$

Since

$$\begin{aligned} \frac{\partial T_{0,n[k]_k}}{\partial q_{i_k}} &= T_{0,(i-1)_k} \frac{\partial (T_{(i-1)_k,i_k}(0) e^{\widehat{s}_{i_k} q_{i_k}})}{\partial q_{i_k}} T_{i_k,n[k]_k} \\ &= T_{0,(i-1)_k} T_{(i-1)_k,i_k}(0) e^{\widehat{s}_{i_k} q_{i_k}} \widehat{s}_{i_k} T_{i_k,n[k]_k} \\ &= T_{0,i_k} \widehat{s}_{i_k} T_{i_k,n[k]_k}, \end{aligned} \tag{4.33}$$

we have

$$dT_{0,n[k]_k} = \sum_{i=1}^{n[k]} T_{0,i_k} \widehat{s}_{i_k} T_{i_k,n[k]_k} dq_{i_k}. \tag{4.34}$$

Left-multiplying both sides of Equation (4.34) by $T^{-1}_{0,n[k]_k}$, then

$$T^{-1}_{0,n[k]_k} dT_{0,n[k]_k} = \sum_{i=1}^{n[k]} T^{-1}_{i_k,n[k]_k} \widehat{s}_{i_k} T_{i_k,n[k]_k} dq_{i_k} \tag{4.35}$$

Equation (4.35) is termed as the differential kinematic equation of branch k. Geometrically, it describes the differential change of the end link pose resulting from the differential changes of joint angles, viewing at the end link frame.

Let $T^d_{0,n[k]_k}$ denote the desired pose of the end link. When it is in the neighborhood of a nominal pose of the end link, denoted by $T_{0,n[k]_k}$, we have

$$dT_{0,n[k]_k} = T^d_{0,n[k]_k} - T_{0,n[k]_k} \tag{4.36}$$

Therefore, the left side of Equation (4.35) becomes

$$T^{-1}_{0,n[k]_k} dT_{0,n[k]_k} = T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k} - I_{4\times 4} \tag{4.37}$$

According to the definition of matrix logarithm [31],

$$\log(T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k}) = T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k} - I - \frac{(T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k} - I)^2}{2} + \frac{(T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k} - I)^3}{3} - \dots \tag{4.38}$$

Using first order approximation, we get

$$T^{-1}_{0,n[k]_k} dT_{0,n[k]_k} = \log\left(T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k}\right) \tag{4.39}$$

Substituting (4.39) into (4.35), we obtain

$$\log\left(T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k}\right) = \sum_{i=1}^{n[k]} T^{-1}_{i_k,n[k]_k} \widehat{s}_{i_k} T_{i_k,n[k]_k} dq_{i_k} \tag{4.40}$$

The explicit formulae for calculating the logarithm of elements of $SO(3)$ and $SE(3)$ are derived by Park [100] (refer to Section 4.2). They are the inverse of exponential map, i.e., log: $SO(3) \mapsto so(3)$ and $SE(3) \mapsto se(3)$. Here, $\log\left(T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k}\right)$ is an element of $se(3)$ so that it can be identified by a 6×1 vector denoted by $\log\left(T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k}\right)^\vee$ in which the first and later three elements represent the position and orientation *differences* between $T_{0,n[k]_k}$ and $T^d_{0,n[k]_k}$. The twist $\widehat{s}_{i_k}$ can also be identified with a 6×1 vector s_{i_k}. Converting Equation (4.40) into the *adjoint representation* [90, 100], we get

$$\log\left(T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k}\right)^\vee = \sum_{i=1}^{n[k]} Ad^{-1}_{T_{i_k,n[k]_k}} s_{i_k} dq_{i_k} \tag{4.41}$$

Since $T_{0,n[k]_k} = T_{0,i_k} T_{i_k,n[k]_k}$, we get $Ad_{T^{-1}_{i_k,n[k]_k}} = Ad_{T^{-1}_{0,n[k]_k}} Ad_{T_{0,i_k}}$. Equation (4.41) can be rewritten as:

$$\log(T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k})^\vee = Ad_{T^{-1}_{0,n[k]_k}} \sum_{i=1}^{n[k]} Ad_{T_{0,i_k}} s_{i_k} dq_{i_k}. \tag{4.42}$$

For simplicity, Equation (4.42) can also be expressed in the following form:

$$D_{T_k} = J_k dq_k, \tag{4.43}$$

where

$D_{T_k} = \log\left(T^{-1}_{0,n[k]_k} T^d_{0,n[k]_k}\right)^\vee \in \Re^{6\times 1}$, is referred to as the *pose difference vector*;

$J_k = A_k B_k S_k \in \Re^{6\times n[k]}$, is termed the *body manipulator Jacobian matrix*;

$A_k = Ad_{T^{-1}_{0,n[k]_k}} \in \Re^{6\times 6}$;
$B_k = row[Ad_{T_{0,1_k}}, Ad_{T_{0,2_k}}, \ldots, Ad_{T_{0,n[k]_k}}] \in \Re^{6\times 6n[k]}$;
$S_k = diag[s_{1_k}, s_{2_k}, \ldots, s_{n[k]_k}] \in \Re^{6n[k]_k \times n[k]}$;
$dq_k = column[dq_{1_k}, dq_{2_k}, \ldots, dq_{n[k]_k}] \in \Re^{n[k]_k \times 1}$.

Equation (4.43) is the differential inverse kinematic equation of a branch. We can get an inverse solution for a desired pose of the end link (an element of $SE(3)$) iteratively by using the Newton-Raphson method with Equation (4.43). However, if the desired pose is not an element of $SE(3)$, rather, a pure orientation (an element of $SO(3)$) or a pure position (a 3×1 vector) problem, the inverse kinematic equation needs to be modified. For this purpose, we express D_{T_k} in a block matrix form as $D_{T_k} = [D_{P_k}, D_{R_k}]^T$ in which $D_{P_k} \in \Re^{3\times 1}$ and $D_{R_k} \in \Re^{3\times 1}$ represent the *position difference vector* and the *orientation difference vector* respectively. Accordingly, $J_{T_k} = [J_{P_k}, J_{R_k}]^T$ in which $J_{P_k} \in \Re^{3\times n[k]}$ and $J_{R_k} \in \Re^{3\times n[k]}$ represent the *body position Jacobian matrix* and the *body orientation Jacobian matrix* respectively. Equation (4.43) becomes

$$\begin{bmatrix} D_{P_k} \\ D_{R_k} \end{bmatrix} = \begin{bmatrix} J_{P_k} \\ J_{R_k} \end{bmatrix} dq_k \tag{4.44}$$

1. **Pure orientation problem**

   A pure orientation problem means that the desired pose of the end link frame is an element of $SO(3)$, i.e., only the frame orientation is to be satisfied. Based on Equation (4.44), we have

   $$D_{R_k} = J_{R_k} dq_k \tag{4.45}$$

   where
   $D_{R_k} = \log\left(R^{-1}_{0,n[k]_k} R^d_{0,n[k]_k}\right)^\vee = \log\left(R^T_{0,n[k]_k} R^d_{0,n[k]_k}\right)^\vee \in \Re^{3\times 1}$;
   $R^d_{0,n[k]_k} \in SO(3)$ is the desired orientation.

2. **Pure position problem**

   A pure position problem for a branch means that the desired pose of the end link frame is a 3×1 position vector, i.e., only the frame position (origin) is to be satisfied. Similar to Equation (4.45), we have

   $$D_{P_k} = J_{P_k} dq_k \tag{4.46}$$

where

$D_{P_k} = R^{-1}_{0,n[k]_k}(P^d_{0,n[k]_k} - P_{0,n[k]_k}) = R^T_{0,n[k]_k}(P^d_{0,n[k]_k} - P_{0,n[k]_k}) \in \Re^{3\times 1}$;

$P^d_{0,n[k]_k} \in \Re^{3\times 1}$ is the desired position.

4.4.2 Differential Kinematics Model for a Tree-structured Robot

A tree-structured robot is a branching type mechanism without any loop. For such a robot, the inverse kinematics problem is to find a set of joint variable values that will place every end-effector to its own desired pose simultaneously. In general, several branches in a tree-structured robot may contain some common links and joints so that the inverse kinematic equations of these branches are dependent on each other. If we treat each of the branches as a serial type robot and individually calculate their inverse solutions, the computation results are usually infeasible because a common joint may be required to take different values at a time in order to place every end link to its own desired pose. Here, we propose an algorithm that can restrict any common joint to take only one value at a time automatically. This algorithm works by orderly combining the inverse kinematic equations of all constituting branches into a single equation according to the *path* matrix. When such an equation is solved numerically, a feasible solution can be derived. The input of this algorithm is a given AIM, a set of desired poses corresponding to all the end links, and an initial guess solution of the joint angles.

Based on the definition of the *path* matrix and Equation (4.43), the combined inverse kinematic equation of a tree-structured robot can be given as follows.

$$D_T = Jdq \tag{4.47}$$

where

$D_T = column[D_{T_1}, D_{T_2}, \ldots, D_{T_m}] \in \Re^{6m\times 1}$, is termed as *generalized pose difference vector*;

$J = ABS \in \Re^{6m\times n}$, is termed as *generalized body manipulator Jacobian matrix*;

$A = diag[A_1, A_2, \ldots, A_m] \in \Re^{6m\times 6m}$;

$$B = \begin{bmatrix} p_{12}Ad_{T_{0,1}} & p_{13}Ad_{T_{0,2}} & \ldots & p_{1(n+1)}Ad_{T_{0,n}} \\ p_{22}Ad_{T_{0,1}} & p_{23}Ad_{T_{0,2}} & \ldots & p_{2(n+1)}Ad_{T_{0,n}} \\ \ldots & & & \\ p_{m2}Ad_{T_{0,1}} & p_{m3}Ad_{T_{0,2}} & \ldots & p_{m(n+1)}Ad_{T_{0,n}} \end{bmatrix} \in \Re^{6m\times 6n}, \text{ in which}$$

p_{ij} $(i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$ is the entry in the ith row and j^{th} column of path matrix $\mathcal{P}(\vec{G})$;

$S = diag[s_1, s_2, \ldots, s_n] \in \Re^{6n \times n}$;

$dq = column[dq_1, dq_2, \ldots, dq_n] \in \Re^{n \times 1}$.

For our purpose, the matrix J can also be written in a block matrix form of $J = column[J_1, J_2, \ldots, J_m]$ with $J_i = column[J_{P_i}, J_{R_i}](i = 1, 2, \ldots, m)$, where J_{P_i} and $J_{R_i} \in \Re^{3 \times n}$.

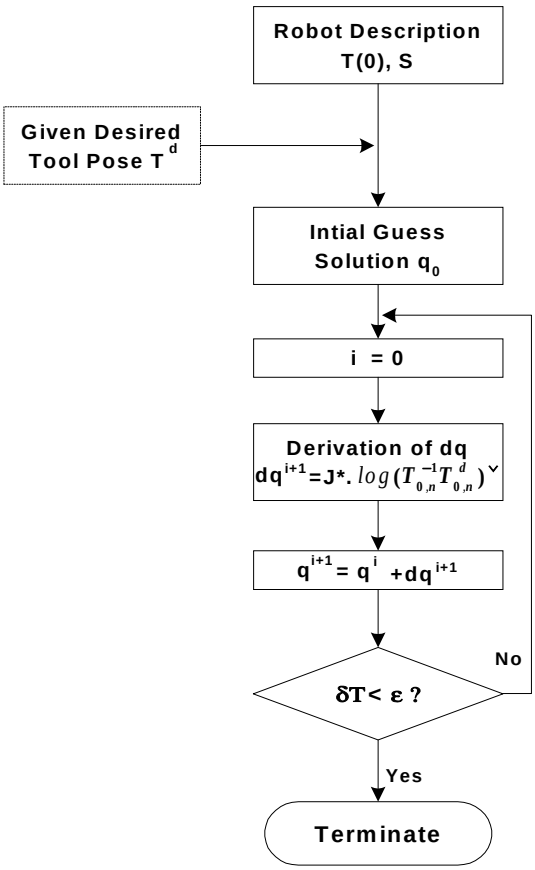


Figure 4.4: Flow chart of iterative inverse kinematics algorithm

Using the Newton-Raphson method, a closed-loop iterative algorithm similar to [66] is employed as shown in Fig. 4.4. This iterative algorithm determines the necessary changes in the joint angles to achieve the differential changes in the poses of the end links. Rewriting Equation (4.47) in an iterative form, we get

$$dq^{i+1} = J^* D_T \tag{4.48}$$

$$q^{i+1} = q^i + dq^{i+1} \tag{4.49}$$

where i represents the number of iterations and J^* is the Moore-Pennose pseudoinverse of J. The singular values decomposition (SVD) method is employed to calculate J^*, which can treat nonsquare and singular Jacobian Matrices in a uniform way [118].

Given an AIM and a set of desired poses T_d corresponding to each of the end links, the algorithm starts from an initial guess, q^0, somewhere in the neighborhood of the desired solution. It is ended when the norm of dq converges to a small tolerance ϵ and the desired poses for every branches thus are achieved simultaneously. Since the AIM contains all the necessary assembly information of a modular robot, the path matrix, forward kinematic equation, and inverse kinematic equation can be generated automatically once an AIM is given. Note that Equation (4.47) works when the desired poses for each of the end link frames are all element of $SE(3)$. However, the inverse kinematics problem of a general tree-structured robot may also be a pure orientation problem, a pure position problem, or a hybrid problem. In order to generalize this algorithm for various inverse kinematics problems, we make the following modifications:

1. **Pure orientation problem**

   The pure orientation problem for a tree-structured robot means that the desired poses of all end links are elements of $SO(3)$, i.e., only the end links' orientations are to be satisfied. Equation (4.47) can be modified as:

   $$D_R = J_R dq \tag{4.50}$$

   where

   $D_R = column[D_{R_1}, D_{R_2}, \ldots, D_{R_m}] \in \Re^{3m\times 1}$, is termed as the *generalized orientation difference vector*;

   $J_R = column[J_{R_1}, J_{R_2}, \ldots, J_{R_m}] \in \Re^{3m\times n}$, is termed as the *generalized body orientation Jacobian matrix.*

2. **Pure position problem**

   The pure position problem for a tree-structured robot means the desired poses of all end link are elements of $\Re^{3\times 1}$, i.e., only the end links' positions are to be satisfied.

Equation (4.47) can be modified as:

$$D_P = J_P dq \tag{4.51}$$

where

$D_P = column[D_{P_1}, D_{P_2}, \ldots, D_{P_m}] \in \Re^{3m\times 1}$, is termed as *generalized position difference vector*;

$J_P = column[J_{P_1}, J_{P_2}, \ldots, J_{P_m}] \in \Re^{3m\times n}$, is termed as *generalized body position Jacobian matrix*.

3. **Hybrid inverse problem**

   The hybrid inverse kinematics problem for a tree-structured robot means that the desired poses set contains either elements of $SE(3)$, $SO(3)$, or $\Re^{3\times 1}$. Hence, the modified inverse kinematic equation should also be a mixed one containing part of either Equation (4.47), (4.50), or (4.51) accordingly. As an example, consider a tree-structure modular robot with three branches. The desired poses for branch 1, branch 2, and branch 3 are elements of $SE(3)$, $SO(3)$, and $\Re^{3\times 1}$ respectively. The resulting inverse kinematic equation for such a hybrid problem can thus be given as follows.

$$D_H = J_H dq \tag{4.52}$$

   where

   $D_H = column[D_{T_1}, D_{R_2}, D_{P_3}] \in \Re^{12\times 1}$,is termed as *hybrid pose difference vector*, and $D_{T_1} \in \Re^{6\times 1}$, D_{R_2}, $D_{P_3} \in \Re^{3\times 1}$;

   $J_H = column[J_1, J_{R_2}, J_{P_3}] \in \Re^{12\times n}$, is termed as *hybrid body manipulator Jacobian matrix*, and $J_1 \in \Re^{6\times n}$, J_{R_2}, $J_{P_3} \in \Re^{3\times n}$, and n is the number of joints;

   $dq = column[dq_1, dq_2, \ldots, dq_n] \in \Re^{n\times 1}$.

In order to illustrate the generality and effectiveness of the proposed inverse kinematics algorithm, two computation examples are presented in this section: one is the inverse kinematics of a tree-structured modular robot; the other is the inverse kinematics of a serial type robot. Note that in these examples, the joint angles are expressed in radians (for revolute joints) or millimeters (for prismatic joints), and the position vectors are in

millimeters. The computation programs are coded in *Mathematica*, and executed on a PC with Pentium 133 processor.

4.4.3 Computation Examples

Example 4.3: As shown in Fig. 4.3, this modular robot has two branches and consists of 11 modules. According to the path matrix, Branch 1 contains modules 0, 1, 2, 3, 4, 6, and 8 in which module 8 is the end link, while Branch 2 contains modules 0, 1, 2, 3, 5, 7, 9, and 10 in which module 10 is the end link. Branch 1 is a 5-DOF (joint 4 is a fixed joint) RPRFRR type open chain, and Branch two is a 7-DOF RPRRRRR type open chain. The common modules of these two branches are modules 0, 1, 2, and 3.

1. The desired poses of module 8 and module 10 are both elements of $SE(3)$. The desired poses are as follows:

   $$T_{0,8} = \begin{bmatrix} -0.5 & -0.5 & -0.707107 & 358.471 \\ -0.5 & -0.5 & 0.707107 & 472.380 \\ -0.707107 & 0.707107 & 0 & 880.546 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and}$$

   $$T_{0,10} = \begin{bmatrix} 0.228553 & 0.780330 & 0.582107 & -22.097 \\ -0.875000 & 0.426777 & -0.228553 & 601.539 \\ -0.426777 & -0.457107 & 0.780330 & 1492.390 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

   Using our inverse kinematics algorithm, the numerical solution of q converges to $[0.7854,\ 100.00,\ 0.7854,\ 0,\ 0.7854,\ 0.7854,\ 0.7854,\ 0.7854,\ 0.7854,\ 0.7854\ ]^T$ after 5 iterations with 5.38 seconds of computation time. The initial guess of q is $[0.5236,\ 75,\ 0.5236,\ 0,\ 0.5236,\ 0.5236,\ 0.5236,\ 0.5236,\ 0.5236,\ 0.5236\ ]^T$. Putting q into the forward kinematics algorithm, the derived end link poses (actual poses) are the same as the desired ones. This indicates that inverse kinematics solution is correct.

2. The desired pose of module 8 is a 3×1 vector, while the desired pose of module 10 is an element of $SO(3)$. The desired poses are as follows:

   $P^d_{0,8} = [358.471, 472.380, 880.546]^T$, and

   $$R^d_{0,10} = \begin{bmatrix} 0.228553 & 0.780330 & 0.582107 \\ -0.875000 & 0.426777 & -0.228553 \\ -0.426777 & -0.457107 & 0.780330 \end{bmatrix}.$$

This is a hybrid inverse kinematics problem. The numerical solution of q converges to [0.7854, 75.00, 0.7918, 0, 0.8745, 0.5236, 0.8914, 0.7175, 0.7046, 0.6917 $]^T$ after 5 iterations with 6.04 seconds of computation time. The initial guess of q is also [0.5236, 75, 0.5236, 0, 0.5236, 0.5236, 0.5236, 0.5236, 0.5236, 0.5236 $]^T$. Using the forward kinematics algorithm, the actual poses corresponding to the numerical result of q are given by

$$T^a_{0,8} = \begin{bmatrix} -0.412101 & -0.578610 & -0.703835 & 358.478 \\ -0.295000 & -0.646139 & 0.703903 & 472.378 \\ -0.862060 & 0.497711 & 0.095585 & 880.549 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and}$$

$$T^d_{0,10} = \begin{bmatrix} 0.228529 & 0.780344 & 0.582097 & -61.974 \\ -0.875005 & 0.426756 & -0.228574 & 615.260 \\ -0.426779 & -0.457102 & 0.780331 & 1448.420 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The actual position of module 8, $P^a_{0,8}$, and the actual orientation of module 10, $R^a_{0,10}$, are also the same as the desired ones respectively.

Example 4.4: As shown in Fig. 4.5, this modular robot consists of 7 modules, sequentially, module 0, 1, 2, 3, 4, 5, and 6 in which module 0 is the base and module 6 is the end link module. It is a 6-DOF RRRRRR type serial open chain. The given AIM and desired pose are

$$\mathcal{A}(G)=\begin{bmatrix} (z,y) & 0 & 0 & 0 & 0 & 0 & L_{r1} \\ (-x,y) & (z,y) & 0 & 0 & 0 & 0 & L_{r1} \\ 0 & (x,y) & (z,-x) & 0 & 0 & 0 & L_{r1} \\ 0 & 0 & (y,x) & (z,y) & 0 & 0 & L_{r1} \\ 0 & 0 & 0 & (x.-z) & (z,y) & 0 & L_{r1} \\ 0 & 0 & 0 & 0 & (-x,y) & (z,y) & L_{r2} \\ 0 & 0 & 0 & 0 & 0 & (x,y) & L_{c2} \\ J_{r1} & J_{r1} & J_{r1} & J_{r1} & J_{r2} & J_{r3} & 0 \end{bmatrix};$$

$$T^d_{0,6} = \begin{bmatrix} -0.426777 & 0.875000 & 0.228553 & -149.226 \\ -0.780330 & -0.228553 & -0.582107 & 115.235 \\ -0.457107 & -0.426777 & 0.780330 & 1164.930 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Actually, the desired pose is derived by putting a set of joint angles q =[0.7854, 0.7854, 0.7854, 0.7854, 0.7854, 0.7854]^T into the forward kinematics algorithm so that the inverse solutions must exist. In order to demonstrate the effectiveness and stability of our algorithm, a comparison with the method proposed by Kelma et al [66] in computational

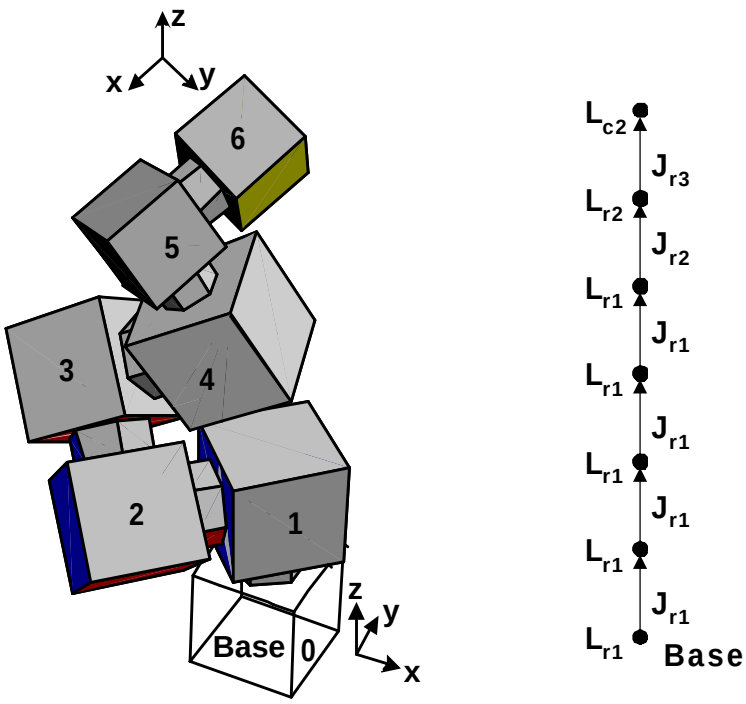


Figure 4.5: A 6-DOF serial robot configuration

aspects is given in Table 4.1. Note that Kelma's algorithm, based on the D-H notation, is also proposed for modular robots. Based on the Newton-Raphson method, this algorithm can also be used to derive the inverse kinematics solutions for serial type robots automatically. It is said to be optimized for computational efficiency and robustness, and has been implemented on a Motorola 68020/68881 based single board computer at rate on the order of 20 Hz.

4.4.4 Remarks on Computation Results

The first computation example is to illustrate the generality of the proposed algorithm. Hence, we choose a more general tree-structured modular robot. This robot, as shown in Fig. 4.3, consists of two branches with three common joints. Each branch contains both revolute joints and prismatic joints. Branch 1 is a 5-DOF nonredundant open chain, while Branch 2 is a 7-DOF redundant one. What encouraged us is that the computation results are satisfactory in both cases, say, only 5 iterations are needed to obtain the desired solutions. This example also shows that the proposed algorithm is a general approach.

Table 4.1: Computation results

Cases	Contents	Algorithm Based on POE Formula	Algorithm Based on D-H Notation
1	Initial Guess	$[0, 0, 0, 0, 0, 0]$	$[0, 0, 0, 0, 0, 0]$
	Iterations	6	17
	Time(Sec.)	2.20	5.98
	Solution	$[-.4456, -.7854, 2.3562, -.4456, .7854, .7854]$	$[.5254, -.5826, 2.3747, -.8794, -.1670, .8787]$
2	Initial Guess	$[0.2, 0.2, 0.2, 0.2, 0.2, 0.2]$	$[0.2, 0.2, 0.2, 0.2, 0.2, 0.2]$
	Iterations	14	96
	Time(Sec.)	4.72	36.85
	Solution	$[.5254, -.5826, 2.3747, -.8794, -.1670, .8787]$	$[.7854, -.4456, 2.3562, -.7854, . - 4456, .7854]$
3	Initial Guess	$[0.4, 0.4, 0.4, 0.4, 0.4, 0.4]$	$[0.4, 0.4, 0.4, 0.4, 0.4, 0.4]$
	Iterations	5	40
	Time(Sec.)	1.87	17.96
	Solution	$[.1827, .5799, .7672, .8783, 1.4156, .8779]$	$[-.4456, -.7854, 2.3562, -.4456, .7854, .7854]$
4	Initial Guess	$[0.6, 0.6, 0.6, 0.6, 0.6, 0.6]$	$[0.6, 0.6, 0.6, 0.6, 0.6, 0.6]$
	Iterations	5	8
	Time(Sec.)	1.76	2.86
	Solution	$[.1827, .5799, .7672, .8783, 1.4156, .8779]$	$[.1827, .5799, .7672, .8783, 1.4156, .8779]$
5	Initial Guess	$[0.75, 0.75, 0.75, 0.75, 0.75, 0.75]$	$[0.75, 0.75, 0.75, 0.75, 0.75, 0.75]$
	Iterations	3	4
	Time(Sec.)	1.1	1.37
	Solution	$[.7854, .7854, .7854, .7854, .7854, .7854]$	$[.7854, .7854, .7854, .7854, .7854, .7854]$
6	Initial Guess	$[0.8, 0.8, 0.8, 0.8, 0.8, 0.8]$	$[0.8, 0.8, 0.8, 0.8, 0.8, 0.8]$
	Iterations	3	3
	Time(Sec.)	0.99	0.93
	Solution	$[.7854, .7854, .7854, .7854, .7854, .7854]$	$[.7854, .7854, .7854, .7854, .7854, .7854]$

It can deal with various inverse kinematics problems of a tree-structured modular robot automatically, regardless of joint types and DOFs.

The second example, on the other hand, is to demonstrate the computational efficiency and stability of this algorithm. As shown in Table 4.1, our algorithm always uses less iterations as well as computation durations than the method proposed in [66] does. As the initial guess solution is given close to the desired solution, the differences between these two algorithms become less distinct because both methods are all locally convergent. This example shows that our algorithm is more efficient and stable even if the initial guess solution is not very close to any nominal solution. The possible reasons are: (1) instead of using the conventional Jacobian matrix, this inverse kinematic algorithm is formulated in terms of the body velocity Jacobian, and (2)without using the differential homogeneous transformation matrix [108], the differential change of the end link pose is computed by using the matrix logarithm on $SE(3)$. As a result, this newly developed algorithm might have a larger local convergent field. However, the exact reasons are still not clear. More research effort needs to be done in the future for an analytical explanation. Note that the algorithm in [66] is also implemented using *Mathematica*.

4.5 Discussion

In this chapter, the kinematic modeling schemes are proposed to generate the forward and inverse kinematics models of a modular robot automatically. Based on the local frame representation of POE formula, the dyad kinematics between two consecutive links is introduced. Applying the dyad kinematics recursively on the adjacent links, the forward kinematics of a tree-structured robot can be derived by means of the path matrix. The inverse kinematics algorithm follows a numerical approach on top of the Newton-Raphson iteration method. It is suitable for both serial type robots and tree-structured ones. It is independent of the number of modules, DOFs, and the types of joints. It can deal with the general inverse kinematics problem (in which the desired configurations are all elements of $SE(3)$), the pure position, pure orientation, and hybrid inverse kinematics problems. However, the inverse kinematics algorithm, like other numerical methods, cannot guarantee that a solution will be derived within a fixed number of iterations. In other words, the computation efficiency strongly depends on the initial guess solution. Hence, we may feel some difficulties when applying this algorithm to on line computations. This algorithm is suitable for tracking applications, in which the initial guess solutions are always very close to the exact solutions [67], and for off-line programming purposes.

Chapter 5

Modular Robot Dynamics

This chapter studies the dynamic issues in modular robots. Similar to the dynamics of conventional robot arms, modular robot dynamics also deals with the formulation of the motion equations of modular robots. The derivation of the dynamic model for a modular robot plays an important role for the simulation of the robot motion, controller design, and the evaluation of the robot configuration design.

For a conventional robot arm, the dynamic equations of motion can be derived manually based on well developed dynamic modeling methods such as the Newton-Euler method, the Kane's method, and the Lagrange method. A modular robot may have very different geometries and DOFs through the reconfiguration of modules. Hence, manual derivation of its dynamic model on a case by case basis needs too much effort. This chapter presents a method to generate the equations of motion of a tree-structured modular robot from a given AIM automatically.

The modeling method presented in this chapter is based on the geometric formulation of the equations of motion for serial type robots addressed by Park *et al* in [103, 104]. We generalize this method for tree-structured modular robots [25, 19, 20]. This method starts with a modified recursive Newton-Euler algorithm formulated entirely in terms of generalized velocities, accelerations, and forces – represented by the elements of $se(3)$. The recursive dynamic equations of a modular robot can thus be derived in terms of standard linear operations on $se(3)$. These equations are then constructed into a closed form by means of the *accessibility* matrix of the kinematic graph of the robot. The closed-form

dynamic model is useful for modular robot design, high level manipulation, and motion optimization.

This chapter is organized as follows. Section 5.1 briefly reviews the significant research efforts in robot dynamics. A geometric formulation of the Newton-Euler equations for an individual link assembly is addressed in Section 5.2. The method that can generate the dynamic model for a tree-structured modular robot automatically is presented in Section 5.3. The inverse and forward dynamic problems are discussed in Section 5.4.

5.1 Introduction

There are two basic issues in modular robot dynamics, i.e., (1) the computation of the actuator torque problem - inverse dynamics, which is to find the actuator forces or torques required to produce a given motion, and (2) the simulation problem - forward dynamics, which is to predict the motion of the robot corresponding to a given time-history of the actuator forces and torques.

Algorithms to solve these two problems can be based on many different general formulations of dynamics such as the Lagrange-Euler (L-E) method [72, 108], the Newton-Euler (N-E) method [6, 81], the Kane's method [61, 63], the recursive Lagrange method [57], and the generalized D'Alembert method [73]. The motion equations in these methods are "equivalent" to each other in the sense that they describe the dynamic behavior of the same physical robot. However, the structure of these equations may differ because they are obtained for various reasons and purposes: to achieve fast computation time in evaluating the nominal joint torques in servoing the robot actuators, to facilitate control analysis and synthesis, and to improve computer simulation of the robot motion. Among these methods, the L-E method and the N-E method have been widely used.

The derivation of the dynamic model of a robot based on the L-E method is simple and systematic. Assuming the robot follows the rigid body motion, the resulting dynamic equations of motion, excluding the dynamics of the electronic control device, gear friction, and backlash, are a set of second-order coupled, nonlinear, differential equations. Each equation contains a large number of torque or force terms such as inertial

torques/forces due to the links, reaction torques/forces generated by acceleration at other joints, velocity-generated (coriolis and centrifugal) reaction torques/forces between joints, and torques/forces due to gravity loading on the links. These torque/force terms depend on the robot's physical parameters and the external load. To improve the speed of computation, simplified sets of equations have been used in [108]. In general, these "approximate" models simplify the underlying physics by neglecting second-order terms such as the coriolis and centrifugal reaction terms. These approximate models, when used for control purpose, restrict the arm movement to low speed. At high speed, the neglected terms become significant, making accurate position control of the robot impossible.

An approach that has the advantage of both computation speed and accuracy is based on the N-E vector formulation. The derivation is simple, but messy, and involves vector cross-product terms. The resulting dynamic equations, excluding the dynamics of the electronic control device, gear friction, and backlash, are a set of forward and backward recursive equations. The forward recursion propagates kinematics information (such as angular and linear velocities and accelerations) from the base frame to the end-effector frame, while the backward recursion propagates the forces and moments exerted on each link from the end-effector frame to the base frame. The equations of motion derived from the N-E method represent the most efficient set of equations of motion of the robot, which makes it possible to achieve real-time control. However, the N-E method is difficult to use for deriving advanced control laws because the recursive equations destroy the "structure" of the dynamic model, which provides useful insight for the controller and robot design.

The geometric formulation of the equations of motion for kinematic chains is a new trend in robot dynamics. By treating $SE(3)$, the special Euclidean group of rigid-body motions as Lie group, this geometric approach can avoid many specific definitions and notations found in the conventional dynamic algorithms. Moreover, it permits a coordinate-free view of robot dynamics, which is more suitable for modular robot applications in which the dynamic algorithm is to be configuration independent. So far, a number of fundamental works have been done by several researchers [12, 90, 103, 104, 102, 117]. Among these works, an unified geometric treatment of robot dynamics is presented by Park *et al* in [103, 104]. In their works, the geometric formulation of the recursive Newton-Euler dynamics

is presented, and followed by the closed-form Lagrangian formulation. The recursive formulation leads to an $O(n)$ algorithm that expresses the dynamics entirely in terms of the coordinate-free Lie algebraic operations. The Lagrangian formulation also expresses the dynamics in terms of these Lie algebraic operations and leads to a particular simple set of closed-form equations. The resulting equations are shown to be computationally effective and to be easily differentiated and factored with respect to any of the robot parameters [104]. These features make the geometric formulation approach very attractive for applications such as robot design, optimal control, and motion optimization.

Following the geometrical approach, this chapter presents a methodology that can generate the dynamic model of a modular robot from a given AIM automatically [25, 19, 20]. The more general tree-structured robots are considered as they are gaining popularity in industry, construction, and multi-legged walking systems [113]. The formulation of the dynamic model is started with a modified recursive Newton-Euler algorithm. The generalized velocities, accelerations, and forces are expressed in terms of linear operations on $se(3)$. Based on the relationship between the recursive formulation and the closed-form Lagrangian formulation for serial robot dynamics discussed in [39, 104, 109], we use an *accessibility* matrix [34] to assist the construction of the closed-form equations of motion of a tree-structured modular robot.

5.2 Newton-Euler Equation for a Link Assembly

As mentioned in Chapter 4, let $\mathrm{v_i}$ and $\mathrm{v_j}$ be two adjacent links connected by joint $\mathrm{e_j}$ as shown in Fig. 4.1 and $\mathrm{v_i}$ precedes $\mathrm{v_j}$. Denote the module coordinate frame on $\mathrm{v_i}$ by frame i, $\mathrm{v_j}$ by frame j. Frame i and j are module frames located at the centroid of the cubes (Fig. 3.6). Note that joint $\mathrm{e_j}$ and link $\mathrm{v_j}$ are rigidly connected together, termed *Link assembly j*.

Now consider two coordinate frames located in link assembly j shown in Fig. 5.1. Link frame j is the module (or local) coordinate frame defined for the modular robot kinematics. The *mass* frame j^* is located at the center of mass of the *entire* link assembly j. Normally, these two frames do not coincide with each other. Let $T_{j^*j} = (p_{j^*j}, R_{j^*j}) \in SE(3)$ be the

pose of link frame j relative to mass frame j^*, where $p_{j^*j} \in \Re^{3\times1}$ and $R_{j^*j} \in SO(3)$ are the position vector and rotation matrix respectively. Note that T_{j^*j} is a constant because frames j and j^* are located on the same rigid link assembly.

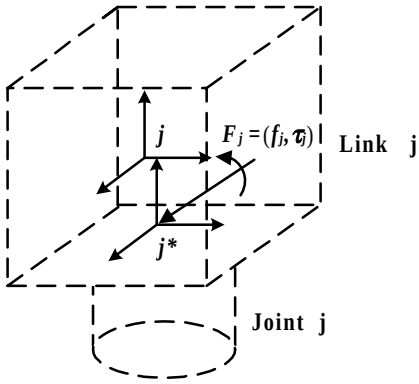


Figure 5.1: Coordinate system in a link assembly

The Newton-Euler equation of this rigid link assembly with respect to its mass frame j^* is given by [90],

$$\mathbf{F}_{j^*} = \begin{bmatrix} f_{j^*} \\ \tau_{j^*} \end{bmatrix} = \begin{bmatrix} m_j I & 0 \\ 0 & J_{j^*} \end{bmatrix} \begin{bmatrix} \dot{v}_{j^*} \\ \dot{w}_{j^*} \end{bmatrix} + \begin{bmatrix} w_{j^*} \times m_j v_{j^*} \\ w_{j^*} \times J_{j^*} w_{j^*} \end{bmatrix} \tag{5.1}$$

Based on the fact that $v_{j^*} \times m_j v_{j^*} = 0$, this equation can be written as

$$\begin{aligned} \mathbf{F}_{j^*} &= \begin{bmatrix} m_j I & 0 \\ 0 & J_{j^*} \end{bmatrix} \begin{bmatrix} \dot{v}_{j^*} \\ \dot{w}_{j^*} \end{bmatrix} \\ &- \begin{bmatrix} -\hat{w}_{j^*} & 0 \\ -\hat{v}_{j^*} & -\hat{w}_{j^*} \end{bmatrix} \begin{bmatrix} m_j I & 0 \\ 0 & J_{j^*} \end{bmatrix} \begin{bmatrix} v_{j^*} \\ w_{j^*} \end{bmatrix} \end{aligned} \tag{5.2}$$

where $\mathbf{F}_{j^*} \in \Re^{6\times1}$ is the resultant wrench applied to the center of mass relative to frame j^*; f_{j^*} and $\tau_{j^*} \in \Re^{3\times1}$ are the total force and moment applied to link assembly j respectively; $M_{j^*} = \begin{bmatrix} m_j & 0 \\ 0 & J_{j^*} \end{bmatrix} \in \Re^{6\times6}$ is the generalized mass matrix expressed in frame j^*. The total masses of link assembly j is m_j (the combined masses of link $\mathrm{v_j}$ and joint $\mathrm{e_j}$). J_{j^*} is the inertia tensor of the link assembly about mass frame j^*. Transforming Equation (5.2) into the *adjoint representation*, we have

$$\mathbf{F}_{j^*} = M_{j^*}\dot{V}_{j^*} - ad^T_{V_{j^*}}(M_{j^*}V_{j^*}). \tag{5.3}$$

The following notations are adopted in Equation (5.3):

- $V_{j^*} = \begin{bmatrix} v_{j^*} \\ w_{j^*} \end{bmatrix} \in \Re^{6\times 1}$ is the generalized body velocity. v_{j^*} and w_{j^*} are 3×1 vectors defining the body translational velocity, $v_{j^*} = (v_x, v_y, v_z)^T$, and the angular velocity, $w_{j^*} = (w_x, w_y, w_z)^T$, respectively.

- $\hat{v}_{j^*}$ and $\hat{w}_{j^*} \in \Re^{3\times 3}$ are skew symmetric matrices related to v_{j^*} and w_{j^*} having the following form:
$\hat{v}_{j^*} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$; $\hat{w}_{j^*} = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}$.

- $ad^T_{V_{j^*}} \in \Re^{6\times 6}$ is the transpose of the adjoint matrix $ad_{V_{j^*}}$ related to V_{j^*} such that

$$ad^T_{V_{j^*}} = (ad_{V_{j^*}})^T = \begin{bmatrix} \hat{w}_{j^*} & \hat{v}_{j^*} \\ 0 & \hat{w}_{j^*} \end{bmatrix}^T = \begin{bmatrix} -\hat{w}_{j^*} & 0 \\ -\hat{v}_{j^*} & -\hat{w}_{j^*} \end{bmatrix}. \tag{5.4}$$

- $\dot{V}_{j^*} = \begin{bmatrix} \dot{v}_{j^*} \\ \dot{w}_{j^*} \end{bmatrix} \in \Re^{6\times 1}$ is the generalized body acceleration.

The Newton-Euler equation, Equation (5.3), is expressed in mass frame j^*. To be consistent with the kinematic model of modular robots described in Chapter 4, this equation must be expressed in the module frame j. According to the wrench transformation method mentioned in [90], the wrench $\mathbf{F}_{j^*}$ can be transformed into an equivalent wrench $\mathbf{F}_j$ expressed in link frame j by

$$\mathbf{F}_j = Ad^T_{T_{j^*,j}} \mathbf{F}_{j^*}, \tag{5.5}$$

where $Ad^T_{T_{j^*,j}}$ is the transpose of $Ad_{T_{j^*,j}}$ such that

$$\begin{aligned} Ad^T_{T_{j^*,j}} &= (Ad_{T_{j^*,j}})^T = \begin{bmatrix} R_{j^*j} & \hat{p}_{j^*j} R_{j^*j} \\ 0 & R_{j^*j} \end{bmatrix}^T \\ &= \begin{bmatrix} R^T_{j^*j} & 0 \\ -R^T_{j^*j}\hat{p}_{j^*j} & R^T_{j^*j} \end{bmatrix}. \end{aligned} \tag{5.6}$$

Note that $\hat{p}_{j^*j}$ is the 3×3 skew symmetric matrix representation of p_{j^*j}. Similarly, the generalized body velocity V_{j^*} and the generalized body acceleration $\dot{V}_{j^*}$ expressed in frame j^* can also be transformed into the equivalent V_j and $\dot{V}_j$ in link frame j as follows.

$$V_{j^*} = Ad_{T_{j^*,j}} V_j \tag{5.7}$$

$$\dot{V}_{j^*} = Ad_{T_{j^*,j}} \dot{V}_j \tag{5.8}$$

Substituting Equations (5.5), (5.7), and (5.8) into (5.3), and using the identity,

$$Ad_{T_{j^*,j}} ad_{V_j} = ad_{Ad_{T_{j^*,j}} V_j} Ad_{T_{j^*,j}} \tag{5.9}$$

we have

$$\mathbf{F}_j = M_j \dot{V}_j - ad^T_{V_j}(M_j V_j) \tag{5.10}$$

This is the Newton-Euler equation of link assembly j written in the link frame j. The generalized mass matrix M_j is relative to link frame j and has the following form:

$$M_j = Ad^T_{T_{j^*,j}} M_{j^*} Ad_{T_{j^*,j}} \tag{5.11}$$

$$= \begin{bmatrix} m_j I & m_j R^T_{j^*j} \hat{p}_{j^*j} R_{j^*j} \\ -m_j R^T_{j^*j} \hat{p}_{j^*j} R_{j^*j} & R^T_{j^*j}(J_{j^*} - m_j \hat{p}^2_{j^*j}) R_{j^*j} \end{bmatrix}. \tag{5.12}$$

For simplicity, we assume that the coordinate axes of mass frame j^* are parallel to the axes of link frame j, then $R_{j^*j} = I$. The generalized mass matrix M_j becomes

$$M_j = \begin{bmatrix} m_j I & m_j \hat{p}_{j^*j} \\ -m_j \hat{p}_{j^*j} & J_{j^*} - m_j \hat{p}^2_{j^*j} \end{bmatrix}. \tag{5.13}$$

Equations (5.3) and (5.10) are of the same form indicating the coordinate-invariant property of Newton-Euler equation in body coordinates. Note that the mass matrix of a link assembly, M_j, can be determined by a computation scheme presented in Appendix A.

5.3 Dynamic Formulation for a Tree-structured Modular Robot

5.3.1 Recursive Newton-Euler Algorithm

The recursive algorithm is a two-step iteration process. For a tree-structured robot, the generalized velocity and acceleration of each link are propagated from the base to the end links of all branches. The generalized force of each link is propagated backward from the end links of all branches to the base. At the branching module, generalized forces transmitted back from all branches are combined.

Forward iteration

The general velocity and acceleration of the base link are given initially,

$$V_b = V_0 = (0,0,0,0,0,0)^T \tag{5.14}$$

$$\dot{V}_b = \dot{V}_0 = (0,0,g,0,0,0)^T \tag{5.15}$$

where V_b and $\dot{V}_b$ are expressed in the base frame 0. We assume that the base frame coincides with the world frame. The gravity effect is taken into consideration by initializing the generalized acceleration (5.15) with the gravitational acceleration constant g. The recursive body velocity and acceleration equations can be written as

$$V_j = Ad_{T_{i,j}^{-1}}(V_i) + s_j\dot{q}_j \tag{5.16}$$

$$\dot{V}_j = Ad_{T_{i,j}^{-1}}(\dot{V}_i) + ad_{Ad_{T_{i,j}^{-1}}(V_i)}(s_j\dot{q}_j) + s_j\ddot{q}_j \tag{5.17}$$

where all the quantities, if not specified, are expressed in link frame j.

- V_j and $\dot{V}_j$ are the generalized velocity and acceleration of link assembly j.
- V_i and $\dot{V}_i$ are the generalized velocity and acceleration of link assembly i written in frame i.
- $\dot{q}_j$ and $\ddot{q}_j$ are the velocity and acceleration of joint $\mathrm{e_j}$ respectively.
- $Ad_{T_{i,j}^{-1}}$ is the *adjoint representation* of $T_{i,j}^{-1}(q_j)$, where $T_{i,j}(q_j) \in SE(3)$ is the pose of frame j relative to frame i with joint displacement q_j.
- $s_j \in \Re^{6\times 1}$ is the twist coordinates of joint $\mathrm{e_j}$, and $s_j = (0,0,0,0,0,0)^T$ for a fixed joint.

Both $Ad_{T_{i,j}^{-1}}(V_i)$ and $s_j\dot{q}_j$ are elements of $se(3)$, and

$$ad_{Ad_{T_{i,j}^{-1}}(V_i)}(s_j\dot{q}_j) = -ad_{s_j\dot{q}_j}(Ad_{T_{i,j}^{-1}}(V_i)). \tag{5.18}$$

In Equation (5.16), we find that

$$Ad_{T_{i,j}^{-1}}(V_i) = V_j - s_j\dot{q}_j. \tag{5.19}$$

Substituting Equation (5.19) into (5.18), we have

$$ad_{Ad_{T_{i,j}^{-1}}(V_i)}(s_j\dot{q}_j) = -ad_{s_j\dot{q}_j}(V_j). \tag{5.20}$$

Therefore, Equation (5.17) can also be written as

$$\dot{V}_j = Ad_{T_{i,j}^{-1}}(\dot{V}_i) - ad_{s_j\dot{q}_j}(V_j) + s_j\ddot{q}_j \tag{5.21}$$

Backward Iteration

The backward iteration of a tree-structured robot starts from all the pendant link assemblies simultaneously. Let $\mathcal{V}_{PD}$ be set of the pendant links of a tree-structured robot. For every pendant link assembly d_i ($\mathrm{v}_{d_i} \in \mathcal{V}_{PD}$), the Newton-Euler Equation (5.10) can be written as

$$F_{d_i} = -F_{d_i}^e + M_{d_i}\dot{V}_{d_i} - ad_{V_{d_i}}^T(M_{d_i}V_{d_i}) \tag{5.22}$$

where F_{d_i} is the wrench exerted on link assembly d_i by its parent (preceding) link relative to frame d_i; $F_{d_i}^e$ is the external wrench exerted on link assembly d_i. Note that the total wrench $\mathbf{F}_{d_i} = F_{d_i} + F_{d_i}^e$.

Now traverse the link assemblies in the tree-structured robot backward from the pendant link assemblies. Let $\mathcal{V}_{Hi}$ be the set of successors of link v_i as viewed from the base to the pendant link assemblies. For every link assembly i, the Newton-Euler Equation (5.10) can be written in the following form:

$$F_i = \sum_{j \in V_{H\,i}} Ad_{T_{i,j}^{-1}}^T(F_j) - F_i^e + M_i\dot{V}_i - ad_{V_i}^T(M_iV_i) \tag{5.23}$$

where all quantities, if not specified, are expressed in link frame i; $F_i \in \Re^{6\times 1}$ is the wrench exerted *to* link assembly i by its ancestor; $F_j \in \Re^{6\times 1}$ is the wrench exerted *by* link assembly i to the successor $\mathrm{v}_j (\in \mathcal{V}_{Hi})$ expressed in link frame j; F_i^e is the external wrench applied to link assembly i. Basically, the total wrench of link assembly i can be written as: $\mathbf{F}_i = F_i - \sum_{j \in V_{H\,i}} Ad_{T_{i,j}^{-1}}^T(F_j) + F_i^e$.

The applied torque/force to link assembly i by the actuator at its input joint $\mathrm{e_i}$ can be calculated by

$$\tau_i = s_i^T F_i \tag{5.24}$$

5.3.2 Closed Form Equations of Motion

By expanding the recursive Newton-Euler Equations (5.14), (5.15),(5.16), (5.21), (5.22), and (5.23) in the (local) module frame coordinates iteratively, we obtain the generalized velocity, generalized acceleration, and generalized force equations in matrix form:

$$\mathbf{V} = \mathbf{HS}\dot{\mathbf{q}} \tag{5.25}$$

$$\dot{\mathbf{V}} = \mathbf{H_{T_0}}\dot{\mathbf{V}}_\mathbf{0} + \mathbf{HS}\ddot{\mathbf{q}} + \mathbf{HA_1V} \tag{5.26}$$

$$\mathbf{F} = \mathbf{H}^T\mathbf{F^E} + \mathbf{H}^T\mathbf{M}\dot{\mathbf{V}} + \mathbf{H}^T\mathbf{A_2MV} \tag{5.27}$$

$$\tau = \mathbf{S}^T\mathbf{F} \tag{5.28}$$

where

$\mathbf{V} = column[V_1, V_2, \cdots, V_n] \in \Re^{6n\times 1}$ is the generalized body velocity vector;

$\dot{\mathbf{V}} = column[\dot{V}_1, \dot{V}_2, \cdots, \dot{V}_n] \in \Re^{6n\times 1}$ is the generalized body acceleration vector;

$\mathbf{F} = column[F_1, F_2, \cdots, F_n] \in \Re^{6n\times 1}$is the body wrench vector;

$\tau = column[\tau_1, \tau_2, \cdots, \tau_n] \in \Re^{n\times 1}$ is the applied joint torque/force vector;

$\dot{\mathbf{q}} = column[\dot{q}_1, \dot{q}_2, \cdots, \dot{q}_n] \in \Re^{n\times 1}$ is the joint velocity vector;

$\ddot{\mathbf{q}} = column[\ddot{q}_1, \ddot{q}_2, \cdots, \ddot{q}_n] \in \Re^{n\times 1}$ is the joint acceleration vector;

$\dot{\mathbf{V}}_\mathbf{0} = (0, 0, g, 0, 0, 0)^T \in \Re^{6\times 1}$ is the generalized acceleration of the base link;

$\mathbf{S} = diag[s_1, s_2, \cdots, s_n] \in \Re^{6n\times n}$ is the joint twist matrix written in the respective body coordinates;

$\mathbf{M} = diag[M_1, M_2, \cdots, M_n] \in \Re^{6n\times 6n}$ is the total generalized mass matrix;

$\mathbf{A_1} = diag[-ad_{s_1\dot{q}_1}, -ad_{s_2\dot{q}_2}, \cdots, -ad_{s_n\dot{q}_n}] \in \Re^{6n\times 6n}$;

$\mathbf{A_2} = diag[-ad^T_{V_1}, -ad^T_{V_2}, \cdots, -ad^T_{V_n}] \in \Re^{6n\times 6n}$;

$\mathbf{F^E} = column[F^e_1, F^e_2, \cdots, F^e_n] \in \Re^{6n\times 1}$ is the external wrench vector;

$$\mathbf{H_{T_0}} = \begin{bmatrix} Ad_{T^{-1}_{01}} \\ Ad_{T^{-1}_{02}} \\ \vdots \\ Ad_{T^{-1}_{0n}} \end{bmatrix} \in \Re^{6n\times 6}$$

$$\mathbf{H} = \begin{bmatrix} I_{6\times 6} & 0 & 0 & \cdots & 0 \\ r_{12}Ad_{T^{-1}_{12}} & I_{6\times 6} & 0 & \cdots & 0 \\ r_{13}Ad_{T^{-1}_{13}} & r_{23}Ad_{T^{-1}_{23}} & I_{6\times 6} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{1n}Ad_{T^{-1}_{1n}} & r_{2n}Ad_{T^{-1}_{2n}} & r_{3n}Ad_{T^{-1}_{3n}} & \cdots & I_{6\times 6} \end{bmatrix} \in \Re^{6n\times 6n}$$

Note that $\mathcal{R}(\vec{G}) = [r_{i,j}] \in \Re^{(n+1)\times(n+1)}$ is the accessibility matrix of the robot's kinematic digraph $\vec{G}$. The matrix $\mathbf{H}$ is termed a *transmission matrix*. The element $r_{i,j}Ad_{T_{i,j}^{-1}}$ relates link frame i and j if link v_i can reach link v_j. Substituting Equations (5.25∼5.27) to (5.28), we obtain the closed-form equations of motion for a tree-structured modular robot with $n+1$ modules (including the base module)

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{N}(\mathbf{q}) = \tau \tag{5.29}$$

where

$$\mathbf{M}(\mathbf{q}) = \mathbf{S}^T\mathbf{H}^T\mathbf{M}\mathbf{H}\mathbf{S} \tag{5.30}$$

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{S}^T\mathbf{H}^T(\mathbf{M}\mathbf{H}\mathbf{A_1} + \mathbf{A_2}\mathbf{M})\mathbf{H}\mathbf{S} \tag{5.31}$$

$$\mathbf{N}(\mathbf{q}) = \mathbf{S}^T\mathbf{H}^T\mathbf{M}\mathbf{H_{T_0}}\dot{\mathbf{V}}_0 + \mathbf{S}^T\mathbf{H}^T\mathbf{F^E} \tag{5.32}$$

$\mathbf{M}(\mathbf{q})$ is the generalized mass matrix of the robot; $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is called *coriolis matrix* for the robot [90]; the vector $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$ gives the coriolis and centrifugal force terms in the equations of motion; $\mathbf{N}(\mathbf{q})$ represents the gravitational force and the external force which act at the joints; τ is the vector of actuator torques/forces.

5.3.3 Remarks on the Dynamics Algorithms

Based on the geometric formulation of the Newton-Euler dynamics equations, both of the recursive and closed-form dynamics algorithms have been presented for general tree-structured modular robots. The major advantage of this approach is that the high level and coordinate-invariant description of robot dynamics can be obtained [109], which facilities our automatic model generation scheme. Regarding the computational complexity (as listed in Table 5.1), the recursive geometric Newton-Euler model is equivalent to a set of efficient $O(n)$ recursive algorithms, while the closed-form dynamic equations of motion are equivalent to a set of $O(n^3)$ algorithms. Note that it is very difficult to estimate the computational complexity of the dynamics algorithms for a tree-structured robot because it depends on not only the number of DOFs of the robot but also the number of branches and the number of modules on each branch. Hence, we use an n-DOF serial type modular robot to estimate the computational complexity of the derived dynamics algorithms as listed in Table 5.1.

Table 5.1: Computational complexity

	Multiplications	Additions
Recursive Algorithm	$342n - 103$	$266n - 174$
Closed-form Algorithm	$48n^3 + 68n^2 + 387n - 108$	$48n^3 + 50n^2 + 317n - 78$

5.3.4 Implementation and Examples

Now, we concentrate on how to generate the closed-form equations of motion automatically, i.e., Equation (5.29), from a given AIM. To formulate equation (5.29), we need to determine the contents of the matrices $\mathbf{M}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ and $\mathbf{N}(\mathbf{q})$.

1. The joint twists in $\mathbf{S}$ are represented in local module frames and are usually parallel to the coordinate axes. They can be obtained from the AIM directly and remain unchanged during the robot motion.

2. The inertia properties of robot modules can be calculated or measured individually, which are listed in Chapter 2. Since the accessibility and path matrices can be derived from the AIM, the generalized mass matrix M_{j^*} (relative to the mass frame j^*) and M_j (relative to the module frame j) of link assembly j can be derived automatically (refer to Appendix A for more details).

3. The forward kinematic algorithms proposed in Chapter 4 and [17] can determine the forward kinematic transformations between any two connective modules in a tree-structured modular robot automatically once the AIM and joint angles are given. Hence, all the adjoint representations in $\mathbf{H}$ and $\mathbf{H}_{\mathbf{T}_0}$ can be calculated. The procedure to obtain the closed-form equations of motion for a modular robot is shown in Fig. 5.2

The dynamic formulation algorithm has been successfully implemented in *Mathematica* code as shown in Fig. 5.2. Once a new modular robot assembly is constructed, a new AIM will be formed and so will the equations of motion of the robot. Note that symbolic expression of the dynamic equations can be obtained as well. Here we give a computation example to illustrate the proposed algorithm.

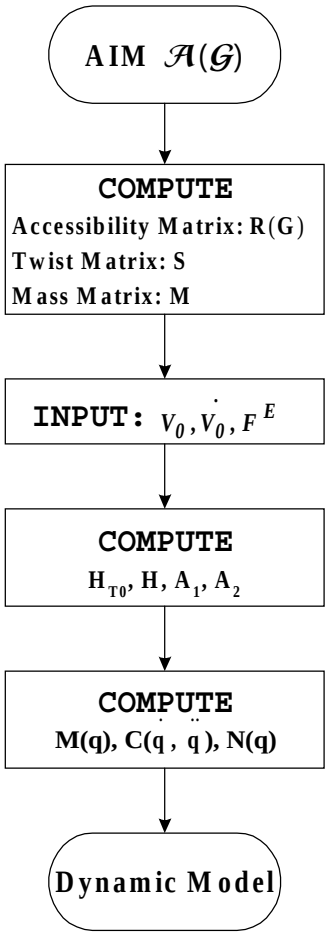


Figure 5.2: Flow chart of dynamic model generation

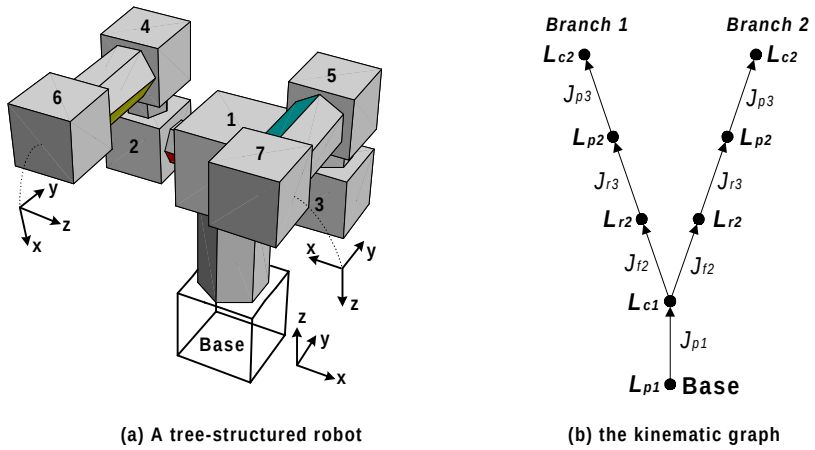


Figure 5.3: A tree-structured modular robot configuration (5-DOF)

Example 5.1: As shown in Fig. 5.3, the modular robot consists of 8 modules and has 2 branches. Its AIM is given as follows.

$$\mathcal{A}(\tilde{G}) = \begin{bmatrix} (z,-y) & 0 & 0 & 0 & 0 & 0 & 0 & L_{p1} \\ (-z,x) & (-y,x) & (y,x) & 0 & 0 & 0 & 0 & L_{c1} \\ 0 & (y,x) & 0 & (z,x) & 0 & 0 & 0 & L_{r2} \\ 0 & 0 & (x,y) & 0 & (z,y) & 0 & 0 & L_{r2} \\ 0 & 0 & 0 & (x,z) & 0 & (z,x) & 0 & L_{p2} \\ 0 & 0 & 0 & 0 & (y,z) & 0 & (z,x) & L_{p2} \\ 0 & 0 & 0 & 0 & 0 & (y,x) & 0 & L_{c2} \\ 0 & 0 & 0 & 0 & 0 & 0 & (y,x) & L_{c2} \\ J_{p1} & J_{f2} & J_{f2} & J_{r3} & J_{r3} & J_{p3} & J_{p3} & 0 \end{bmatrix}.$$

Based on the given AIM, the accessibility and path matrices of the kinematic graph shown in Fig. 5.3 can be given by

$$\mathcal{R}(\vec{G}) = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix};$$

$$\mathcal{P}(\vec{G}) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Here we denote the joint angles by $\mathbf{q} = (q_1, q_2, \ldots, q_7)^T$, joint velocity by $\dot{\mathbf{q}} = (\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_7)^T$, and joint acceleration by $\ddot{\mathbf{q}} = (\ddot{q}_1, \ddot{q}_2, \ldots, \ddot{q}_7)^T$. Note that since joints 2 and 3 are both fixed joints, $q_2 = q_3 = 0$; $\dot{q}_2 = \dot{q}_3 = 0$; and $\ddot{q}_2 = \ddot{q}_3 = 0$. Also, assume that the generalized external forces applied on the link assemblies are all equal to zero. According to the given AIM as well as the physical parameters defined on links and joints, the proposed algorithm can automatically generate a set of closed-form dynamics equations for this

modular robot at an arbitrary posture (determined by $q, \dot{q}, \ddot{q}$) as follows.

$$
\begin{aligned}
\tau_1 &= 494.424 + 50.4\ddot{q}_1 \\
\tau_2 &= 0 \\
\tau_3 &= 0 \\
\tau_4 &= 0.7\ddot{q}_4 + 2.912q_6\ddot{q}_4 + 4.05q_6^2\ddot{q}_4 + 2.912\dot{q}_4\dot{q}_6 + 8.1q_6\dot{q}_4\dot{q}_6 \\
\tau_5 &= 0.7\ddot{q}_5 + 2.912q_7\ddot{q}_5 + 4.05q_7^2\ddot{q}_5 + 2.912\dot{q}_5\dot{q}_7 + 8.1q_7\dot{q}_5\dot{q}_7 \\
\tau_6 &= 4.05\ddot{q}_6 - 1.456\dot{q}_4^2 - 4.05q_6\dot{q}_4^2 \\
\tau_7 &= 4.05\ddot{q}_7 - 1.456\dot{q}_5^2 - 4.05q_7\dot{q}_5^2
\end{aligned}
$$

Because the given robot configuration is not very complicated, we can still manually formulate its dynamic model based on the traditional Newton-Euler method. It has shown that the manually derived dynamic equations are identical to the equations listed above. Note that SI units are used in the derived equations.

5.4 Inverse and Forward Dynamics Problem

5.4.1 Inverse Dynamics

The inverse dynamics problem is to determine the joint torques $\tau(t)$ which are needed to generate the motion specified by the joint accelerations $\ddot{\mathbf{q}}(t)$, velocities $\dot{\mathbf{q}}(t)$, and angles $\mathbf{q}(t)$ when the external forces $\mathbf{F}^E$ are known.

Solving the inverse dynamics problem is necessary for robot motion planning and control algorithm implementation. Once a robot trajectory is specified in terms of joint angles, velocities, and accelerations, and the external forces are also given, the inverse dynamics allows the computation of the torques to be applied to the joints to obtain the desired motion. This computation turns out to be useful for verifying the feasibility of the imposed trajectory and for compensating nonlinear terms in the dynamic model of a robot. Obviously, the computation of inverse dynamics becomes quite straightforward once the dynamic model is derived.

5.4.2 Forward Dynamics

The forward dynamics problem is to determine the joint accelerations $\ddot{\mathbf{q}}(t)$, velocities $\dot{\mathbf{q}}(t)$, and angles $\mathbf{q}(t)$ resulting from the given joint torques/forces τ and the external forces $\mathbf{F}^E$ when the initial joint angles $\mathbf{q}(t_0)$ and velocities $\dot{\mathbf{q}}(t_0)$ are known.

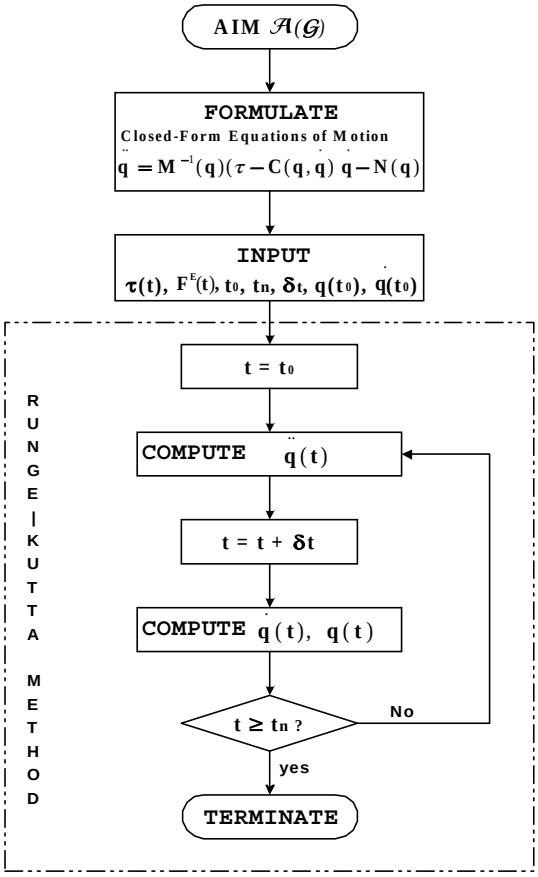


Figure 5.4: Flow chart of the forward dynamic algorithm

Solving the forward dynamics problem is necessary for the robot simulation. Forward dynamics allows describing the motion of the real physical system in terms of the joint accelerations, when a set of assigned joint torques is applied to the robot. The joint velocities and angles can be obtained by integrating the system of nonlinear differential equations. The closed-form equations of motion derived form the proposed algorithm is more convenient for this purpose because they provide the explicit relationship between the joint torques (as well as the external forces), angles, velocities, and accelerations.

Equation (5.29) can also be rewritten as

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q})(\tau - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{N}(\mathbf{q})). \tag{5.33}$$

Equation (5.33) consists of a set of second order differential equations. For the simulation of robot motion, the initial state at the time instant t_0 is known with the joint angles $\mathbf{q}(t_0)$ and velocities $\dot{\mathbf{q}}(t_0)$. The forward dynamic problem then solves a set of second order differential equations subjected to the given initial conditions. Many numerical integration methods can be used to derive the solutions. In our forward dynamic algorithm, the fourth order Runge-Kutta method is employed for solutions. The inputs of this algorithm include a given AIM, the initial time instant t_0, the ending time instant t_n, the integration step δt, the initial joint angles $\mathbf{q}(t_0)$, the joint velocities $\dot{\mathbf{q}}(t_0)$, the applied joint torques $\tau(t)$, and the external forces exerted on the each of the link assemblies $\mathbf{F}^{\mathbf{E}}(t)$, where $t_0 \leq t \leq t_n$. The outputs are the joint accelerations, velocities, and angles at different time instants. The flow chart of the forward kinematic algorithm is shown in Fig. 5.4. In order to illustrate this algorithm, a computation example is given as follows.

Example 5.2: In this example, we still use the same robot configuration as well as the AIM as given in Example 5.1. Suppose that $\mathbf{q}(t_0)$, $\dot{\mathbf{q}}(t_0)$, and $\mathbf{F}^{\mathbf{E}}(t)$ are all equal to zero; $\tau(t)$ takes a constant value such that $\tau(t)$=(500,0,0,-0.5,0.5,0.05,0.05)^T; and $t_0 = 0$, $t_n = 1.5$s, $\delta t = 0.05$s. The selected computation results are given in Table 5.2, 5.3, and 5.4, and Fig. 5.5, 5.6, and 5.7. The SI units are used in the computation results. The corresponding robot configurations at different time instants are shown in Fig. 5.8.

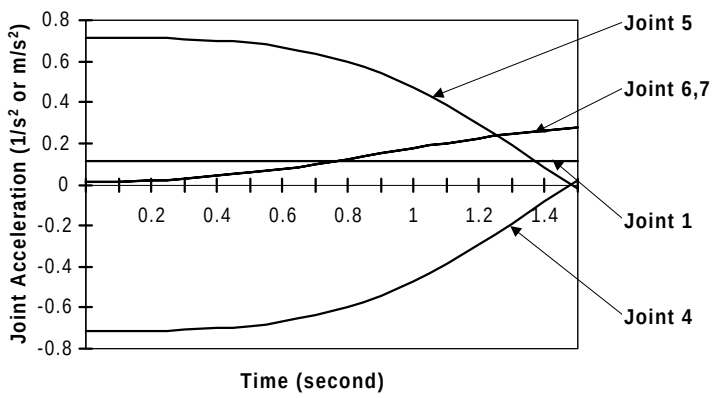


Figure 5.5: Joint acceleration

Table 5.2: Joint acceleration ($1/s^2$ or m/s^2)

Time(s)	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7
0	0.1106	0	0	-0.7147	0.7147	0.0123	0.0123
0.3	0.1106	0	0	-0.7080	0.7080	0.0288	0.0288
0.6	0.1106	0	0	-0.6667	0.6667	0.0769	0.0769
0.9	0.1106	0	0	-0.5392	0.5392	0.1494	0.1494
1.2	0.1106	0	0	-0.2913	0.2913	0.2261	0.2261
1.5	0.1106	0	0	0.0193	-0.0193	0.2771	0.2771

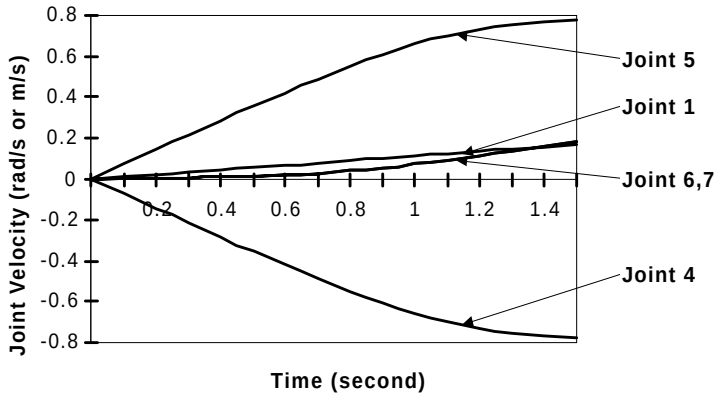


Figure 5.6: Joint velocity

Table 5.3: Joint velocity ($1/s$ or m/s)

Time(s)	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7
0	0	0	0	0	0	0	0
0.3	0.0332	0	0	-0.2138	0.2138	0.0054	0.0054
0.6	0.0663	0	0	-0.4215	0.4215	0.0205	0.0205
0.9	0.0996	0	0	-0.6052	0.6052	0.0540	0.0540
1.2	0.1328	0	0	-0.7326	0.7326	0.1106	0.1106
1.5	0.1660	0	0	-0.7734	0.7734	0.1871	0.1871

Table 5.4: Joint displacement (rad or m)

Time(s)	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7
0	0	0	0	0	0	0	0
0.3	0.0050	0	0	-0.0321	0.0321	0.0007	0.0007
0.6	0.0199	0	0	-0.1277	0.1277	0.0042	0.0042
0.9	0.0448	0	0	-0.2827	0.2827	0.0148	0.0148
1.2	0.0797	0	0	-0.4852	0.4852	0.0389	0.0389
1.5	0.1245	0	0	-0.7134	0.7134	0.0832	0.0832

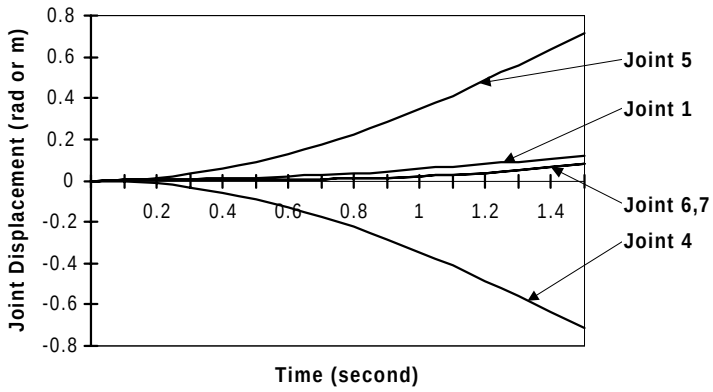


Figure 5.7: Joint displacement

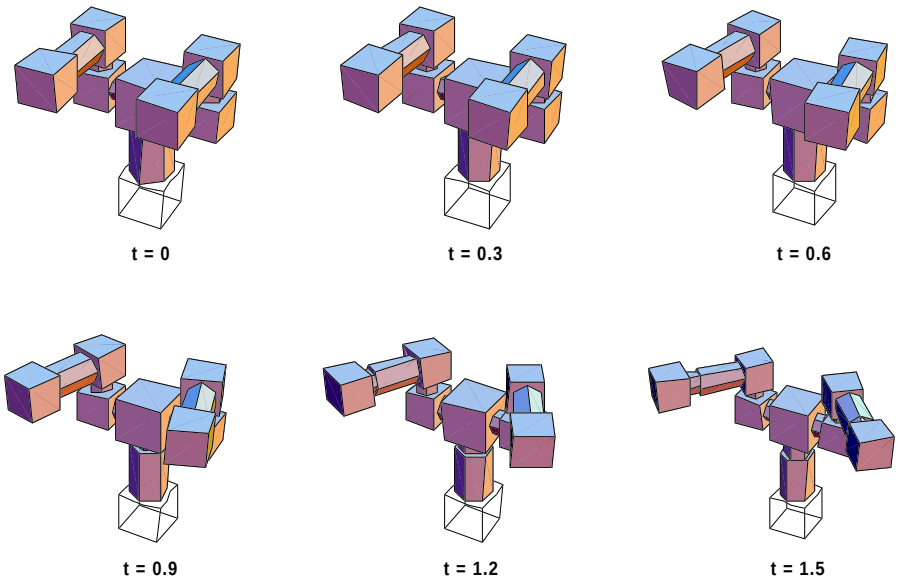


Figure 5.8: Robot postures at different time instants

5.5 Discussion

In general, the dynamics of a robot can be formulated in either recursive Newton-Euler or closed-form Lagrangian style. The recursive Newton-Euler method is computationally efficient. However, many applications still rely on explicit closed-form expression of the equations of motion. Since a modular robot has unfixed configurations, it is not intuitive and advisable to derive the closed-form dynamic model manually. Following the geometric modeling approach, this chapter introduces a dynamic formulation scheme that can generate both the recursive and the closed-form dynamic models of a tree-structured modular robot from a given AIM automatically. The formulation of dynamic models starts with the modified recursive Newton-Euler algorithm in which the generalized velocities, accelerations, and forces (wrenches) are expressed in terms of linear operations on $se(3)$. The recursive dynamic model is then constructed into a closed form by means of the accessibility matrix. As compared with the conventional dynamic algorithms and the general dynamic simulation software such as DADS and Working Model, the proposed algorithm provides a high-level description of robot dynamics, and has proved to be simple and convenient for modular robot applications. Computationally, the recursive geometric Newton-Euler model is equivalent to a set of $O(n)$ recursive algorithms, while the closed-form dynamic equations of motion are equivalent to a set of $O(n^3)$ algorithms.

Chapter 6

Kinematic Calibration for Modular Robots

This chapter addresses the kinematic calibration of modular robots. The purpose of robot kinematic calibration is to compensate for the kinematic errors in the robot (due to machining tolerance, assembly errors, and wear and drift of mechanical parts) and to improve its positioning accuracy. Different from a conventional robot which has a monolithic design, a modular robot can be reconfigured into various configurations to suit a diversity of tasks. As a result, assembly errors are likely to be introduced to a modular robot. Kinematic calibration therefore is important for a modular robot system.

In this chapter, a novel kinematic calibration algorithm is formulated based on the local frame representation of POE formula. The basic idea of the proposed algorithm is to redefine a set of new local coordinate frames to describe the kinematics of the robot precisely. In order to simplify the calibration model, these new frames are defined in such a way that makes the joint twists and joint angles retain their nominal values. All the geometric errors are therefore lumped into the initial poses of the consecutive modules. Because of the use of local coordinates, this algorithm is easy to implement and can deal with robots of general tree-structured geometry regardless of the number of DOFs and types of joints.

This chapter is organized as follows. Section 6.1 is a general introduction about robot calibration. The calibration models are introduced in Section 6.2. Section 6.3 provides three computation examples to demonstrate the accuracy, effectiveness, and generality of

this approach.

6.1 Introduction

In the modular robot system, a set of modules is joined together through connectors/adapters to form a complete robot assembly. However, due to the machining tolerance in the fabrication of the modules, the misalignments in the consecutive modules during assembly, and the wear and drift in the mechanical parts after a long period of use, geometric errors are introduced in the robot kinematic parameters. This will certainly lower the positioning accuracy of the robot. Compared to a monolithic designed industrial robot, a modular robot has more connecting points. Furthermore, its configuration will often be changed to cater for different task requirements through reselection and reconfiguration of the modules. Hence, identifying the critical kinematic parameters to improve the positioning accuracy of the end-effector is a very important issue for modular reconfigurable robots. Modular robot calibration comes into play in at least two situations. One is after the reconfiguration of a robot geometry. In this case, new kinematic parameters of the robot have to be identified and corrected. The other is after a large number of operations because of the possible loose module connections. The actual kinematic parameters may vary from those previously calibrated.

Basically, kinematic calibration is necessary for all kinds of robots. In recent years, there has been an explosion in the literature on the topic of kinematic calibration, reflecting its current importance and development in robotics. Many calibration models are presented by researchers who are motivated by the need for robot calibration [9, 59, 89, 112, 119]. Based on the different kinematic modeling techniques used, we categorized them into (1)the Denavit-Hartenberg (D-H) parameterization approach [53, 54, 55, 56, 119, 128, 131], (2)the Zero Reference Position modeling approach [64, 65, 87, 88], (3)the Continuous and Parametrically Complete (CPC) modeling approach [142, 143, 144], and (4)the Product-of-Exponentials (POE) formulation approach [21, 26, 91, 106, 122, 123, 136, 137].

The use of the D-H representation [32] for kinematics is the de-facto standard in robotics. This modeling technique intelligently uses a minimal set of parameters to describe the

relationship between adjacent joint axes for the formulation of the kinematics. However, being adopted for calibration modeling, this method is not amenable to direct identification as all the parameters in this kinematic model are stringently defined and are unique to the particular robot configuration concerned [119]. In addition, it has the singularity problem when neighboring joint axes are nearly parallel. For example, one of the parameters in the D-H model is the length of the normal between two consecutive joint axes. If two axes are parallel, an infinite number of common normals exist with all having the same length. However, when one of the axes becomes slightly misaligned (with two axes still in a same plane), the length of this common normal will instantaneously jump to zero. This variation of kinematic parameters with respect to changes in joint axis being not continuous, may result in severe numerical difficulties during the parameter identification of the calibration process [112]. A number of researchers use modified forms of the D-H formulation to overcome the singularity problem. Hayati *et al.* [53, 54, 55, 56] introduce an angular alignment parameter β_i in place of the d_i parameter in the D-H model to represent a small misalignment between two consecutive parallel axes, and an additional linear parameter for handling prismatic joints. Many researchers [60, 62, 120, 126] pursue their research based on Hayati's modified D-H model. Stone [119] uses a six-parameter representation (S-model) in which two additional parameters are added to the D-H model to allow for arbitrary placement of link frames.

Mooring [87], Mooring and Tang [88], and Kazerounian and Qian [64, 65] follow a so-called "zero-reference" position method approach as described in [121, 48]. This method describes the robot kinematics in terms of the axes directions and locations in the zero reference position. Construction of a calibration model based on this method is not very straightforward. The elimination of redundant parameters in [88] requires an assumption that the rotation and translation errors in certain directions are negligible. However, this technique is not prone to the difficulties encountered in case of parallel or nearly parallel joint axes when using the D-H notations.

Zhuang *et al.* [142, 143, 144] proposes a six-parameter CPC model. In this model, a singularity-free line representation consisting of four line parameters is adopted to ensure parameterically continuous, while two additional parameters are used to allow arbitrary

placement of the link coordinate frames to make the model complete.

The POE formula is uniform in modeling robots with both revolute and prismatic joints, and provides a very structured parameterization for an open chain robot. Significantly, the kinematic parameters in the POE formula vary smoothly with changes in joint axes so that no special descriptions are required when adjacent joint axes are close to parallel. This fact makes POE formula more suitable for calibration purpose. Park and Okamura [91, 106] propose a calibration model based on the POE formula for general open chain robots. This model assumes that kinematic errors exist in each of the joint axes and the home position of the tool frame. All the kinematic parameters are expressed with respect to the base frame, and the attachment of local frames to each joint axis is unnecessary.

Because the robot geometries that a modular robot system can generate differ significantly, e.g., serial and tree-structured types, conventional calibration schemes dealing with specific type of robots with serial geometry are not suitable under such circumstances. Moreover, because of the modularity, it is simple and convenient to express the kinematic parameters in the local reference frames. Following the POE formula approach, we propose here a singularity-free kinematic calibration modeling method for the modular reconfigurable robot system [21, 122, 123, 136, 137]. This calibration model is based on the local frame representation of the POE formula, termed the *Local POE Model* or the *L-model*. In this modeling scheme, instead of identifying the kinematic errors in the joint axes relative to the module frames, which may involve complicated computations, all the kinematic errors are assumed to exist in the relative initial pose of every dyad. The calibration procedure is to redefine a new local coordinate frame on each module, in which the twist coordinates of the joint axis and joint angles retain their nominal values. By linearizing the forward kinematic equation with respect to the kinematic parameters in the local coordinate frames, four compact error models are derived for robot calibration. Then an iterative least-squares algorithm is employed for calibration solution. Three simulation examples of calibrating a 3-module robot, a SCARA type robot, and a tree-structured robot are demonstrated. The simulation results have shown that the average positioning accuracy of the end-effector increases significantly after calibration.

6.2 Kinematic Calibration Models

In general, the kinematic calibration model, i.e., the error model, can be derived by linearizing the forward kinematic equation with respect to the kinematic parameters. The kinematic calibration thus involves making several measurements of the end-effector pose error and determining the optimal (actual) values of the kinematic parameters to minimize the error in the least-square sense. For convenience, we first formulate the calibration model for serial type robots, and then modify the model to suit the calibration of tree-structured robots.

Consider a serial type robot with $n+1$ modules and an end-effector (tool). Without loss of generality, we sequentially number the local coordinate frames from 0 (representing the base frame) to $n+1$ (representing the tool frame). The forward kinematics can be given by

$$\begin{aligned} T_{0,n+1}(q) &= T_{0,1}(q_1)T_{1,2}(q_2)\ldots T_{n-1,n}(q_n)T_{n,n+1} \\ &= T_{0,1}(0)e^{\hat{s}_1 q_1}T_{1,2}(0)e^{\hat{s}_2 q_2}\ldots T_{n-1,n}(0)e^{\hat{s}_n q_n}T_{n,n+1} \end{aligned} \tag{6.1}$$

where $T_{n,n+1}$ represents the fixed kinematic transformation form the end module (link) frame to the tool frame. The assembly of the end module (end link) and the tool can be considered as a fixed kinematic dyad, termed as the *last dyad*.

According to Equation (6.1), the nominal forward kinematics $T_{0,n+1}(q)$ or simply $T_{0,n+1}$ is a function of $T(0)$, S, and q, denoted by $T = f(T(0), S, q)$, where $T(0)$, S, and q represent the initial poses, twist coordinates, and joint angles respectively, i.e., $T(0) = [T_{0,1}(0)$, $T_{1,2}(0)$,..., $T_{n-1,n}(0)$, $T_{n,n+1}]^T$, $S = [s_1, s_2,\ldots,s_n]^T$, and $q = [q_1, q_2,\ldots,q_n]^T$.

We first consider two consecutive modules $(i-1)$ and i, i.e., a dyad, as shown in Fig. 6.1. Due to the geometric errors, module i's nominal frame i and twist $\hat{s}_i$ are shifted to the actual frame 'i^a' and twist '$\hat{s}_i^a$' respectively. The kinematic calibration, therefore, becomes to identify each of the actual module frames, joint twists, and the joint angle offsets. Of course, we can formulate an error model to directly identify these kinematic quantities. However, such a model will be very complicated because every dyad contains 13 redundant and highly coupled error parameters, i.e., six for the actual frame, six for the actual twist,

and one for the actual joint angle. Especially, to identify the actual twists (joint axes) is rather difficult and may involve complicated computations [106].

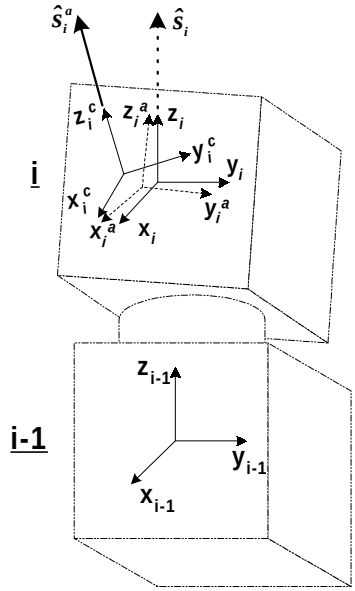


Figure 6.1: Coordinate frames in a dyad

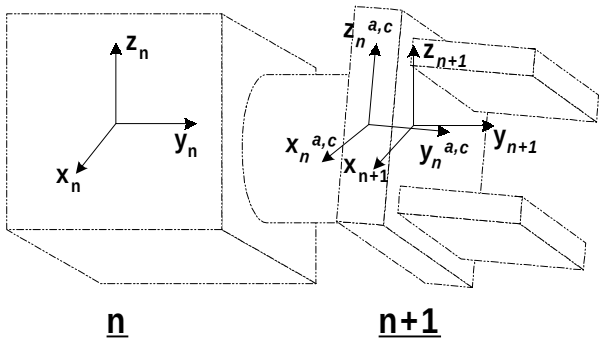


Figure 6.2: Coordinate frames in the end-effector

By taking advantage of the local POE formula in which the local reference frames can be arbitrarily assigned, we are able to redefine a new local frame, termed the calibrated frame 'i^c' for each of modules during the calibration process. Then the robot positioning errors are minimized in the least-square sense, maintaining the twists and joint angles at their nominal values (Fig. 6.1) in the new defined local frames. As a result, the calibrated frame i^c may not coincide with the actual frame i^a. In this case, the new kinematic errors, termed the modeling errors, will be introduced to the dyad. Fortunately, the modeling errors in a dyad can be lumped into its succeeding dyad.

In the 'last dyad', i.e., the last module and the tool as shown in Fig. 6.2, there is no joint twist so that the geometric errors only exist in the initial pose. We, therefore, are able to recover the modeling errors presented in its preceding dyad and also achieve the coincidence of both the calibrated tool frame $(n+1)^c$ and actual (measured) tool frame $(n+1)^a$ upon calibration. If we calibrate a robot without a tool, we can still add a tool frame that normally coincides with the last module frame. This additional dyad, termed as the *compensation dyad*, can be used to compensate for the modeling errors presented in its preceding dyad. Without loss of generality, we also assume that the base model does not have geometric errors, i.e., the nominal and actual base frames coincide with the world frame. Therefore, the kinematic calibration becomes to identify the new defined local frames so as to reflect the robot's actual characteristics.

6.2.1 Basic Calibration Models

According to the definition of matrix logarithm defined on $SE(3)$, we can find at least a $\hat{p} \in se(3)$ for a given $T \in SE(3)$, such that $e^{\hat{p}} = T$. Hence, it is sufficient to let $e^{\hat{p}_i} = T_{i-1,i}(0)$ (with $e^{\hat{p}_{n+1}} = T_{n,n+1}$), where $\hat{p}_i \in se(3)$ $(i = 1, 2, \ldots, n+1)$. Equation (6.1) can be rewritten as:

$$T_{0,n+1} = e^{\hat{p}_1} e^{\hat{s}_1 q_1} e^{\hat{p}_2} e^{\hat{s}_2 q_2} \ldots e^{\hat{p}_n} e^{\hat{s}_n q_n} e^{\hat{p}_{n+1}}. \tag{6.2}$$

Assume that the kinematic errors occur only in $\hat{p}_i$, hence in $T_{i-1,i}(0)$ $(i = 1, 2, \ldots, n)$ and $T_{n,n+1}$. By linearizing the forward kinematic equation, Equation (6.2) with respect to $\hat{p}_i$, the error models for robot calibration can be derived. Because the kinematic errors in $\hat{p}_i$ can be expressed in either frame i or frame $i-1$, the following two basic calibration models can be formulated accordingly.

1. Let the kinematic errors in $\hat{p}_i$ be expressed in module frame i, denoted by $d\hat{p}_{i,i}$ or simply $d\hat{p}_i$. Since $\hat{p}_i \in se(3)$, the error $d\hat{p}_i$ is also an element of $se(3)$. Geometrically, $d\hat{p}_i = e^{-\hat{p}_i} d(e^{\hat{p}_i})$ (refers to Appendix B) so that $d(e^{\hat{p}_i}) = e^{\hat{p}_i} d\hat{p}_i$. Linearizing Equation (6.2) with respect to $\hat{p}_i$, we have

$$\begin{aligned} dT_{0,n+1} &= d(e^{\hat{p}_1}) e^{\hat{s}_1 q_1} e^{\hat{p}_2} e^{\hat{s}_2 q_2} \ldots e^{\hat{p}_n} e^{\hat{s}_n q_n} e^{\hat{p}_{n+1}} \\ &\quad + e^{\hat{p}_1} e^{\hat{s}_1 q_1} d(e^{\hat{p}_2}) e^{\hat{s}_2 q_2} \ldots e^{\hat{p}_n} e^{\hat{s}_n q_n} e^{\hat{p}_{n+1}} \end{aligned}$$

$$
\begin{aligned}
& + \ldots + e^{\hat{p}_1} e^{\hat{s}_1 q_1} \ldots e^{\hat{p}_{n-1}} e^{\hat{s}_{n-1} q_{n-1}} d(e^{\hat{p}_n}) e^{\hat{s}_n q_n} e^{\hat{p}_{n+1}} \\
& + e^{\hat{p}_1} e^{\hat{s}_1 q_1} \ldots e^{\hat{p}_n} e^{\hat{s}_n q_n} d(e^{\hat{p}_{n+1}}) \\
= \; & e^{\hat{p}_1} d\hat{p}_1 e^{\hat{s}_1 q_1} e^{\hat{p}_2} e^{\hat{s}_2 q_2} \ldots e^{\hat{p}_n} e^{\hat{s}_n q_n} e^{\hat{p}_{n+1}} \\
& + e^{\hat{p}_1} e^{\hat{s}_1 q_1} e^{\hat{p}_2} d\hat{p}_2 e^{\hat{s}_2 q_2} \ldots e^{\hat{p}_n} e^{\hat{s}_n q_n} e^{\hat{p}_{n+1}} \\
& + \ldots + e^{\hat{p}_1} e^{\hat{s}_1 q_1} \ldots e^{\hat{p}_{n-1}} e^{\hat{s}_{n-1} q_{n-1}} e^{\hat{p}_n} d\hat{p}_n e^{\hat{s}_n q_n} e^{\hat{p}_{n+1}} \\
& + e^{\hat{p}_1} e^{\hat{s}_1 q_1} \ldots e^{\hat{p}_n} e^{\hat{s}_n q_n} e^{\hat{p}_{n+1}} d\hat{p}_{n+1}. \qquad (6.3)
\end{aligned}
$$

Right-multiplying both sides of the resulting equation by $T_{0,n+1}^{-1}$ and using the *adjoint representation*, the following error model can be shown:

$$
\begin{aligned}
dT_{0,n+1} T_{0,n+1}^{-1} \; = \; & Ad_{e^{\hat{p}_1}} d\hat{p}_1 + Ad_{e^{\hat{p}_1} e^{\hat{s}_1 q_1}} Ad_{e^{\hat{p}_2}} d\hat{p}_2 \\
& + \ldots + Ad_{e^{\hat{p}_1} e^{\hat{s}_1 q_1} e^{\hat{p}_2} e^{\hat{s}_2 q_2} \ldots e^{\hat{p}_n} e^{\hat{s}_n q_n}} Ad_{e^{\hat{p}_{n+1}}} d\hat{p}_{n+1}, \qquad (6.4)
\end{aligned}
$$

where $dT_{0,n+1} T_{0,n+1}^{-1} \in se(3)$ represents the gross kinematic errors of a complete robot with respect to the base frame. According to the formulation of matrix logarithm as mentioned in Chapter 4, $dT_{0,n+1} T_{0,n+1}^{-1}$ can be given by

$$
dT_{0,n+1} T_{0,n+1}^{-1} = \log(T_{0,n+1}^{a} T_{0,n+1}^{-1}), \qquad (6.5)
$$

where $T_{0,n+1}^{a}$ represents the actual (measured) pose of the tool frame with respect to the base frame. Since elements of $se(3)$ can also be identified by 6-dimensional vectors Through: $d\hat{p}_i \mapsto dp_i = (dx_i, dy_i, dz_i, \delta x_i, \delta y_i, \delta z_i)^T \in \Re^{6\times 1}$ and $\log(T_{0,n+1}^{a} T_{0,n+1}^{-1})$ $\mapsto \log(T_{0,n+1}^{a} T_{0,n+1}^{-1})^{\vee} = (dx_{0,n+1},\, dy_{0,n+1},\, dz_{0,n+1},\, \delta x_{0,n+1},\, \delta y_{0,n+1},\, \delta z_{0,n+1})^T \in \Re^{6\times 1}$, Equation (6.4) can be expressed as:

$$
\begin{aligned}
\log(T_{0,n+1}^{a} T_{0,n+1}^{-1})^{\vee} \; = \; & Ad_{T_{0,1}(0)} dp_1 + Ad_{T_{0,1}} Ad_{T_{1,2}(0)} dp_2 \\
& + \ldots + Ad_{T_{0,n}} Ad_{T_{n,n+1}(0)} dp_{n+1}, \qquad (6.6)
\end{aligned}
$$

where $T_{0,i} = e^{\hat{p}_1} e^{\hat{s}_1 q_1} e^{\hat{p}_2} e^{\hat{s}_2 q_2} \ldots e^{\hat{p}_i} e^{\hat{s}_i q_i}$, representing the forward kinematic transformation from the base frame 0 to module frame i. After dp_i is identified, the calibrated initial pose of module frame i with respect to module frame $i-1$, $T_{i-1,i}^{c}(0)$, can be given by

$$
T_{i-1,i}^{c}(0) = e^{\hat{p}_i} e^{d\hat{p}_i} = T_{i-1,i}(0) e^{d\hat{p}_i} \qquad (6.7)
$$

2. Let the kinematic errors in $\hat{p}_i$ be expressed in the module frame $i-1$, denoted by $d\hat{p}_{i-1,i}$. Then $d\hat{p}_{i-1,i} \in se(3)$ and $d\hat{p}_{i-1,i} = d(e^{\hat{p}_i})e^{-\hat{p}_i} = Ad_{T_{i-1,1}(0)} d\hat{p}_i$. Equation (6.6) can be rewritten as:

$$\log(T^a_{0,n+1} T^{-1}_{0,n+1})^\vee = dp_{0,1} + Ad_{T_{0,1}} dp_{1,2} + \ldots + Ad_{T_{0,n}} dp_{n,n+1}, \tag{6.8}$$

where $dp_{i-1,i} = (dx_{i-1,i},\, dy_{i-1,i},\, dz_{i-1,i},\, \delta x_{i-1,i},\, \delta y_{i-1,i},\, \delta z_{i-1,i})^T \in \Re^{6\times 1}$. After $dp_{i-1,i}$ is identified, the calibrated initial pose of module frame i with respect to module frame $i-1$, $T^c_{i-1,i}(0)$, can be given by

$$T^c_{i-1,i}(0) = e^{d\hat{p}_i} e^{\hat{p}_i} = e^{d\hat{p}_i} T_{i-1,i}(0). \tag{6.9}$$

The basic error models of Equation (6.6) and (6.9) are derived by observing the gross kinematic errors of a complete robot with respect to the base frame. Alternatively, the gross kinematic errors of the robot can also be expressed with respect to the tool frame. Left-multiplying both sides of Equation (6.6) and (6.9) by $Ad_{T^{-1}_{0,n+1}}$, the following two equations can be easily shown:

$$\begin{aligned} \log(T^{-1}_{0,n+1} T^a_{0,n+1})^\vee &= Ad_{T^{-1}_{0,n+1}} (Ad_{T_{0,1}(0)} dp_1 + Ad_{T_{0,1}} Ad_{T_{1,2}(0)} dp_2 \\ &\quad + \ldots + Ad_{T_{0,n}} Ad_{T_{n,n+1}(0)} dp_{n+1}); \end{aligned} \tag{6.10}$$

$$\log(T^{-1}_{0,n+1} T^a_{0,n+1})^\vee = Ad_{T^{-1}_{0,n+1}} (dp_{0,1} + Ad_{T_{0,1}} dp_{1,2} + \ldots + Ad_{T_{0,n}} dp_{n,n+1}); \tag{6.11}$$

where $\log(T^{-1}_{0,n+1} T^a_{0,n+1})^\vee$ is also a 6-dimensional vector associated with $\log(T^{-1}_{0,n+1} T^a_{0,n+1}) \in se(3)$, representing the gross kinematic errors of the robot with respect to the tool frame, and $Ad_{T^{-1}_{0,n+1}} \log(T^a_{0,n+1} T^{-1}_{0,n+1}) = \log(T^{-1}_{0,n+1} T^a_{0,n+1}) = T^{-1}_{0,n+1} dT_{0,n+1}$.

By expressing the local and gross kinematic errors of a robot with respect to different reference frames, four compact calibration models are formulated, i.e., Equation (6.6), (6.9), (6.10), and (6.11). For the purpose of kinematic calibration, these models are equivalent. From the computational point of view, however, these four models are slightly different. For example, the term $\log(T^{-1}_{0,n+1} T^a_{0,n+1})^\vee$ in both Equation (6.10) and (6.11) can be calculated in a simple and efficient way such that:

$$\log(T^{-1}_{0,n+1} T^a_{0,n+1})^\vee = \begin{bmatrix} (R - R^T)/2 & R^T_{0,n+1}(P^a_{0,n+1} - P_{0,n+1}) \\ 0 & 0 \end{bmatrix}, \tag{6.12}$$

where $T^a_{0,n+1}$ and $T_{0,n+1}$ are given by $(P^a_{0,n+1}, R^a_{0,n+1})$ and $(P_{0,n+1}, R_{0,n+1})$ respectively. In Equation (6.12), $R = R^T_{0,n+1} R^a_{0,n+1}$. Geometrical interpretations of the kinematic terms such as $d\hat{p}_i$, $d\hat{p}_{i-1,i}$, $d(e^{\hat{p}_i})e^{-\hat{p}_i}$, $e^{-\hat{p}_i}d(e^{\hat{p}_i})$, $dT_{0,n+1}T^{-1}_{0,n+1}$, and $T^{-1}_{0,n+1}dT_{0,n+1}$, can be found in Appendix B.

6.2.2 An Iterative Least-Squares Algorithm

Based on the calibration models, an iterative least-squares algorithm is employed for the calibration solution. This algorithm is suitable for each of the calibration models. For simplicity, we will introduce this algorithm according to the error model of Equation (6.6).

Apparently, Equation (6.6) can also be expressed as a linear equation of the form

$$y = Ax, \tag{6.13}$$

where

$y = \log(T^a_{0,n+1}T^{-1}_{0,n+1})^\vee \in \Re^{6\times 1}$

$x = [dp_1, dp_2, \ldots, dp_{n+1}]^T \in \Re^{6(n+1)\times 1}$

$A = \left[Ad_{T_{0,1}(0)}, Ad_{T_{0,1}}Ad_{T_{1,2}(0)}, \ldots, Ad_{T_{0,n}}Ad_{T_{n,n+1}(0)}\right] \in \Re^{6\times 6(n+1)}$.

$T^{-1}_{0,n+1}$ and A can be obtained from the nominal model, while $T^a_{0,n+1}$ can be obtained from the actual measurement data. Hence, $y = \log(T^a_{0,n+1}T^{-1}_{0,n+1})^\vee$ can be computed. x is the kinematic errors to be identified. It reflects the errors in the robot.

The model presented above identifies six kinematic error parameters to calibrate the initial pose of each dyad. An n-DOF modular robot will totally have $6(n+1)$ error parameters to be identified. This number is greater than that of the conventional calibration model in which the number of error parameters is equal to $4R + 2P$, where R and P represent the number of revolute and prismatic joints respectively [89]. Consequently, the A matrix in Equation (6.13) is larger than that of the convention model, which will make the computation heavy . Since the calibration computation is usually an off-line computation scheme and the A matrix, in general, is not very large, the computation complexity of the calibration models becomes less important.

The identification of the kinematic errors requires comparing the difference between the robot's actual (measured) and nominal poses. To improve the calibration accuracy, we usually need to measure the tool frame in many different robot postures. Suppose we need to take m measured pose data. For i^{th} measurement, we obtain a y_i. The corresponding A_i can be determined from the nominal kinematic model. After m measurements, we can stack y_i and A_i to form the following equation:

$$\widetilde{Y} = \widetilde{A}x, \tag{6.14}$$

where
$\widetilde{Y} = [y_1, y_2, \ldots, y_m]^T \in \Re^{6m\times 1}$,
$x = [dp_1, dp_2, \ldots, dp_{n+1}]^T \in \Re^{6(n+1)\times 1}$,
$\widetilde{A} = column[A_1, A_2, \ldots, A_m] \in \Re^{6m\times 6(n+1)}$.

Obviously, Equation (6.14) is a linear calibration model. Since the model consists of $6m$ linear equations with $6(n+1)$ variables ($m > (n+1)$), the linear least-squares algorithm is employed for the parameter identification. The least-square solution of x is given by

$$x \;=\; (\widetilde{A}^T\widetilde{A})^{-1}\widetilde{A}^T\widetilde{y} \tag{6.15}$$

where $(\widetilde{A}^T\widetilde{A})^{-1}\widetilde{A}^T$ is the pseudoinverse of $\widetilde{A}$.

The solution of Equation (6.15) can be further improved through iterative substitution as shown in Fig 6.3. Let $T(0)$ denote the vector of the initial poses of the dyads and initialize $T(0)$ using the nominal value, i.e, $T(0) = [T_{0,1}(0),\ T_{1,2}(0),\ \ldots, T_{n-1,n}(0),\ T_{n,n+1}]^T$. Based on the measured data and Equation (6.15), we can obtain the kinematic error parameter vector, x. The vector $T(0)$ is updated by substituting x into Equation (6.7). The same procedure is repeated until the norm of the error vector, $||x||$, approaches zero and the vector $T(0)$ converges to some stable values. Then the final $T(0)$ represents the calibrated initial poses of the dyads, i.e., $T(0) = [T_{0,1}(0)^c,\ T_{1,2}(0)^c, \ldots, T_{n,n+1}(0)^c]^T$.

Note that the kinematic error vector, x, will no longer represent the actual kinematic errors in the dyads after iterations. However, the actual kinematic errors in each of the dyads can be extracted by comparing $T_{i-1,i}(0)$ with $T_{i-1,i}(0)^c$. Based on the matrix

logarithm, Equation (6.7) can also be written as

$$dp_i = \log(T_{i-1,i}^{-1}(0)T_{i-1,i}^{c}(0))^{\vee}. \tag{6.16}$$

After calibration, the forward kinematic equation becomes

$$T_{0,n+1} = T_{0,1}^{c}(0)e^{\hat{s}_1 q_1}T_{1,2}^{c}(0)e^{\hat{s}_2 q_2}\ldots T_{n-1,n}^{c}(0)e^{\hat{s}_n q_n}T_{n,n+1}^{c} \tag{6.17}$$

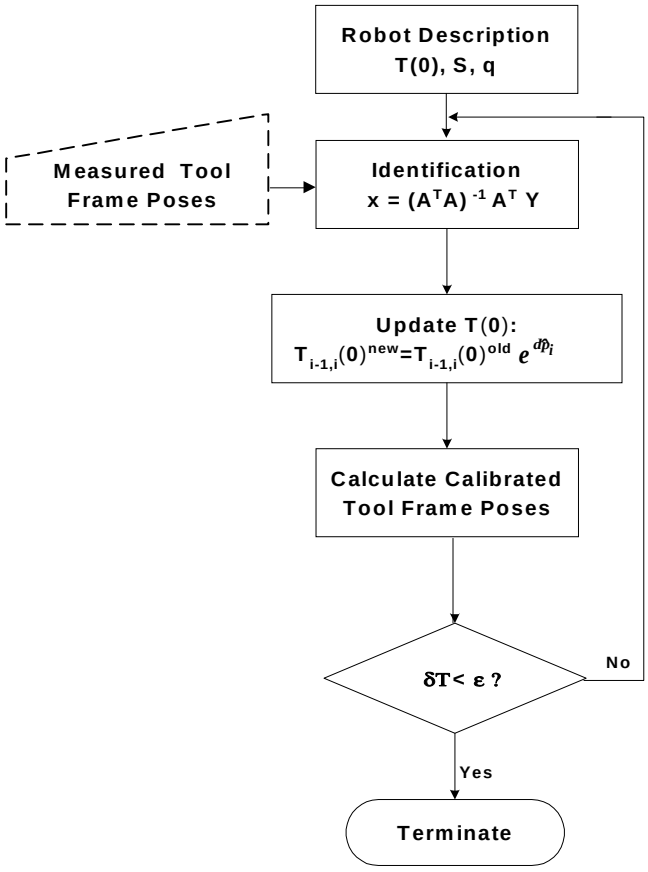


Figure 6.3: Iterative calibration algorithm

6.2.3 Kinematic Calibration of Tree-structured Robots

All the calibration models mentioned above are based on the local POE formula. They can be easily modified to suit the calibration of a tree-structured modular robot. Since the branches in a tree-structured modular robot may contain several common modules (or dyads), all these branches are to be calibrated simultaneously in order to get a unique calibrated initial pose for each of the common dyads. Similar to the modeling scheme

for inverse kinematics, once the path matrix is derived from a given AIM, the calibration model can be automatically constructed.

Now we consider a tree-structured modular robot consisting of $n+1$ modules and m branches. The path matrix of such a robot therefore has m rows and $(n+1)$ columns. Identical to the labeling scheme adopted in Section 4.3, we assume that row k ($k = 1, 2, \ldots, m$) has $n[k]+1$ nonzero entries, sequentially 0_k or simply 0 (because of the unique base), $1_k, \ldots, n[k]_k$ from the left to right. Here i_k is the index of the ith non-zero entry in row k, and $i = 0, 1, \ldots, n[k]$. Since the tools that will be rigidly attached to the pendant links are not considered in the AIM and the path matrix, the last dyad or the compensation dyad will be added to each branch. Based on Equation (6.14), the calibration equation for branch k can be given by:

$$Y_k \;=\; A_k x_k, \tag{6.18}$$

where

$Y_k = \log(T^a_{0,n[k]+1} T^{-1}_{0,n[k]+1})^\vee \in \Re^{6\times 1}$

$x_k = \left[dp_{1_k}, dp_{2_k}, \ldots, dp_{n[k]_k}, dp_{(n[k]+1)_k}\right]^T \in \Re^{6(n[k]+1)\times 1}$

$A_k = \left[Ad_{T_{0,1_k}}(0), Ad_{T_{0,1_k}} Ad_{T_{1_k,2_k}}(0), \ldots, Ad_{T_{0,(n[k]-1)_k}} Ad_{T_{(n[k]-1)_k,n[k]_k}}(0), Ad_{T_{0,n[k]_k}}\right] \in \Re^{6\times 6(n[k]+1)}$.

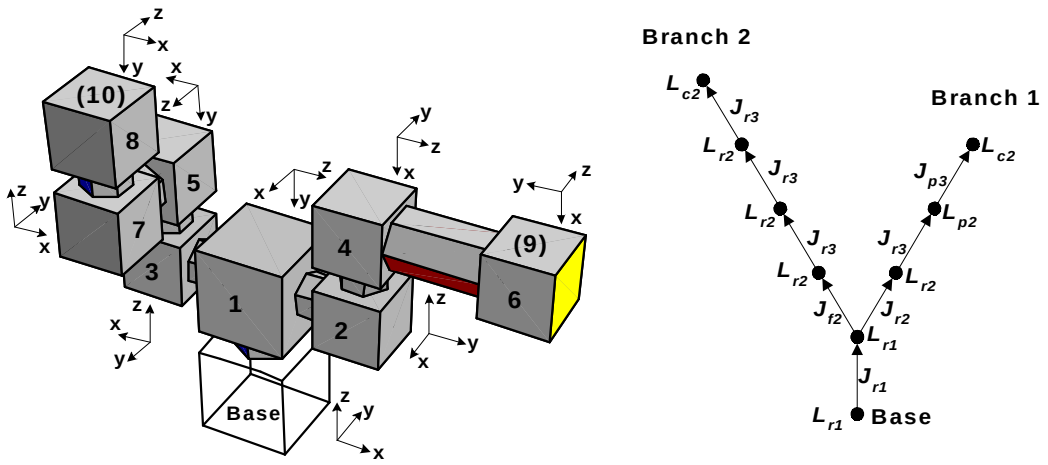


Figure 6.4: A tree-structured modular robot configuration (7-DOF)

The formulation of the calibration model for a tree-structured modular robot is similar to that of the inverse kinematics model, i.e., arranging the calibration equations of all branches into a single equation according to the path matrix. Instead of presenting

the formulation details, we use an example to show the arranging scheme. As shown in Fig. 6.4, the tree-structured modular robot consists of 9 modules (7-DOFs). The AIM and path matrix are given as follows.

$$\mathcal{A}(\tilde{G}) = \begin{bmatrix} (z,-y) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & L_{r1} \\ (y,x) & (z,x) & (-z,x) & 0 & 0 & 0 & 0 & 0 & L_{r1} \\ 0 & (-y,x) & 0 & (z,x) & 0 & 0 & 0 & 0 & L_{r2} \\ 0 & 0 & (-x,y) & 0 & (z,y) & 0 & 0 & 0 & L_{r2} \\ 0 & 0 & 0 & (x,-y) & 0 & (z,x) & 0 & 0 & L_{p2} \\ 0 & 0 & 0 & 0 & (y,z) & 0 & (z,x) & 0 & L_{r2} \\ 0 & 0 & 0 & 0 & 0 & (y,x) & 0 & 0 & L_{c2} \\ 0 & 0 & 0 & 0 & 0 & 0 & (y,-x) & (z,x) & L_{r2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (y,x) & L_{c2} \\ J_{r1} & J_{r2} & J_{f2} & J_{r3} & J_{r3} & J_{p3} & J_{r3} & J_{r3} & 0 \end{bmatrix};$$

$$\mathcal{P}(\vec{G}) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

According to the path matrix, we know that this robot has two branches. The first branch consists of link modules 0,1,2,4 and 6; the second branch consists of link modules 0,1,3,5,7, and 8. Since the tools or end-effectors are not attached to each of the pendant links, a compensation dyad will be added to each branch. Based on the path matrix $\mathcal{P}(\vec{G})$ and Equation (6.18), the calibration model for this robot can be written as:

$$Y = Ax, \tag{6.19}$$

where
$Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} \log(T^a_{0,9}T^{-1}_{0,9})^\vee \\ \log(T^a_{0,10}T^{-1}_{0,10})^\vee \end{bmatrix} \in \Re^{12\times 1}$;
$x = [dp_1, dp_2, \ldots, dp_8, dp_9, dp_{10}]^T \in \Re^{60\times 1}$;
$A = \begin{bmatrix} B_{0,1} & A_{0,1}B_{1,2} & 0 & A_{0,2}B_{2,4} & 0 & A_{0,4}B_{4,6} & 0 & 0 & A_{0,6} & 0 \\ B_{0,1} & 0 & A_{0,1}B_{1,3} & 0 & A_{0,3}B_{3,5} & 0 & A_{0,5}B_{5,7} & A_{0,7}B_{7,8} & 0 & A_{0,8} \end{bmatrix} \in \Re^{12\times 60}$, in which $A_{i,j} = Ad_{T_{i,j}}$ and $B_{i,j} = Ad_{T_{i,j}(0)}$.

This example shows that the proposed calibration models are also capable of calibrating robots with tree-structured geometry that most of the multi-arm robots belong to. Therefore, the local POE calibration model is a completely general approach.

6.3 Computation Examples

This section provides three computation examples to demonstrate the convergence, effectiveness, and generality of the proposed calibration algorithm. The first example is a 3-module (2-DOF) serial robot. Because of the simple configuration, the modeling strategy can be clearly demonstrated. The second example is a 7-module (4-DOF) SCARA type robot. The purpose of this example is to demonstrate that the proposed calibration algorithm does not suffer from the singularity problem when two neighboring joint axes are close to parallel. The last example is a general tree-structured modular robot consisting of different types of joints. It is used to show the generality of the algorithm. Note that, all of the computation results are based on the calibration model of Equation (6.6). In these examples, we consider modular robots without any end-effector or tool.

6.3.1 Calibration of a 3-module Robot

As shown in Fig. 6.5, the given modular robot consists of two large revolute joint modules and one small cubic link module. Its AIM is given as follows.

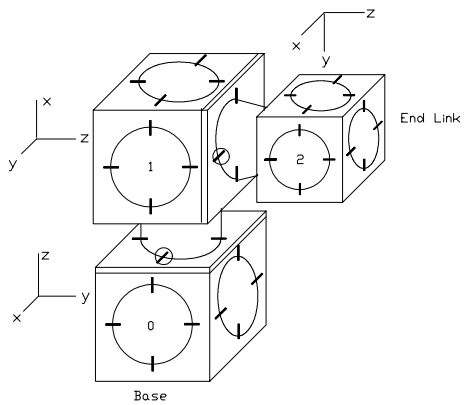


Figure 6.5: A 3-module robot configuration (2-DOF)

$$\mathcal{A}(\tilde{G}) = \begin{bmatrix} (z,x) & 0 & L_{r1} \\ (-x,y) & (z,y) & L_{r1} \\ 0 & (-z,x) & L_{c2} \\ J_{r1} & J_{r2} & 0 \end{bmatrix}.$$

According to the AIM, we find that:

$s_1 = (0,\ 0,\ 0,\ 1,\ 0,\ 0)^T$; $s_2 = (0,\ 0,\ 0,\ 0,\ 0,\ 1)^T$;

$$T_{0,1}(0) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 350 \\ 0 & 0 & 0 & 1 \end{bmatrix}; T_{1,2}(0) = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 312.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note that all the angles are expressed in radians, and the lengths are in millimeters. To validate the proposed calibration models, we present here a simulated calibration result. The measured (simulated) end link frame poses are obtained from Equation (6.1) (with $T_{n,n+1} = I$) under a set of preset geometric errors. This procedure is stated as follows.

- Assign the geometric errors

  dp_i, dq_i, and ds_i (i = 1, 2) (listed in Table 6.1)

- Find the actual initial pose in each dyad

  $T^a_{i-1,i}(0)$= $T_{i-1,i}(0)e^{d\hat{p}_i}$(i = 1, 2)

- Calculate the simulated end link frame poses

  $T^a_{0,2}(q_1 + dq_1, q_2 + dq_2)$= $T^a_{0,1}(0)e^{(\hat{s}_1+d\hat{s}_1)(q_1+dq_1)}T^a_{1,2}(0)e^{(\hat{s}_2+d\hat{s}_2)(q_2+dq_2)}$

In this example, the number of measured poses is set to 10, i.e, $m = 10$. The calibration results are given in Table 6.2, 6.3, and Fig. 6.6. Table 6.2 shows the identified kinematic errors in the dyads. Because there is no end-effetor or tool in this robot, the compensation term is also identified and given in the last row of Table 6.2. It is apparent that the preset geometric errors and the kinematic errors to be identified have different physical meaning and are not one-to-one correspondent so that the preset geometric errors can not be fully recovered. Nevertheless, as shown in Table 6.3, two end link poses (arbitrarily selected) after calibration are the same as the measured ones. It follows that these two sets of error parameters are equivalent in the sense of describing the actual end link poses. In robot calibration, the positioning accuracy is the main concern. Whether the error parameters can be fully recovered becomes less important. Fig. 6.6 also shows that the calibrated (updated nominal) end link poses $T_{0,n} = (p_{0,n}, R_{0,n})$ fully converge to the measured ones $T^a_{0,n} = (p^a_{0,n}, R^a_{0,n})$ after only a few iterations. Note that the position and orientation deviations are given by $dp = \sqrt{\frac{\sum_{i=1}^{m}\|p^a_{0,n}-p_{0,n}\|}{m}}$ and $dR = \sqrt{\frac{\sum_{i=1}^{m}\|\log({R^a_{0,n}}^{-1}R_{0,n})^\vee\|}{m}}$, where dp

and dR can be interpreted as the average position and orientation differences between the measured (actual) end link poses and the calibrated end link poses for m measurements respectively. The computation results indicate that the proposed algorithm is correct and effective.

Table 6.1: Preset geometric errors for the 3-module robot

	dp_i	ds_i	dq_i
Dyad 1	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 0, 2., -0.0002, 0.02, 0)^T$	0.02
Dyad 2	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(2., 0, 0, 0, 0.02, -0.0002)^T$	0.02

Table 6.2: Identified kinematic errors for the 3-module robot

	dp_i
Dyad 1	$(-0.961, -0.073, 2.010, 0.0272, 0.0201, 0.0397)^T$
Dyad 2	$(0.009, 0.939, 1.013, 0.0001, 0.0072, 0.0270)^T$
Compensation Dyad	$(-0.007, -1.990, 1.033, 0.0200, -0.0001, -0.0070)^T$

Table 6.3: End link poses before and after calibration (3-module robot)

Joint Angle	$q = (.9948, -1.5184)^T$	$q = (2.8798, 2.8798)^T$
Nominal Pose	$\begin{bmatrix} .0285 & .5439 & -.8387 & -262.085 \\ .0439 & .8375 & .5446 & 170.200 \\ .9986 & -.0523 & 0 & 350.000 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} .9330 & .2500 & -.2588 & -80.881 \\ -.2500 & -.0670 & -.9659 & -301.852 \\ -.2588 & .9659 & 0 & 350.000 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Measured Pose	$\begin{bmatrix} .0780 & .5063 & -.8588 & -269.263 \\ .0419 & .8590 & .5102 & 164.279 \\ .9961 & -.0758 & .0458 & 362.220 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} .9322 & .2662 & -.2450 & -73.885 \\ -.2273 & -.0957 & -.9691 & -303.688 \\ -.2815 & .9591 & -.0286 & 343.134 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Calibrated Pose	$\begin{bmatrix} .0780 & .5063 & -.8588 & -269.263 \\ .0419 & .8590 & .5102 & 164.279 \\ .9961 & -.0758 & .0458 & 362.220 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} .9322 & .2662 & -.2450 & -73.885 \\ -.2273 & -.0957 & -.9691 & -303.688 \\ -.2815 & .9591 & -.0286 & 343.134 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Now we consider whether the actual twists expressed in the calibrated module frames retain their nominal coordinates as we expected. The coordinates of the actual twists expressed in the local actual module frames can be given by:

$s_1^a = s_1 + ds_1 = (0, 0, 2., .9998, .02, 0)^T$ and

$s_2^a = s_2 + ds_2 = (2., 0, 0, 0, .02, .9998)^T$.

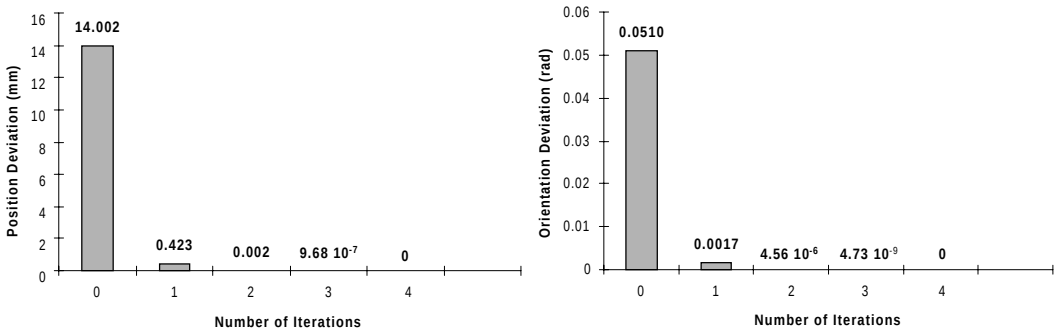


Figure 6.6: Calibration convergence (3-module robot)

According to the preset errors, the actual module frames with respect to the base frame can be computed as

$$T^a_{0,1}(0) = \begin{bmatrix} .0200 & .9990 & -.0396 & 2.000 \\ -.0204 & .0400 & .9990 & 2.040 \\ .9996 & -.0192 & .0212 & 352.001 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

$$T^a_{0,2}(0) = \begin{bmatrix} .9980 & -.0603 & -.0184 & -8.448 \\ .0208 & .0392 & .9990 & 316.345 \\ -.0596 & -.9974 & .0403 & 356.662 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

From the calibration results, the calibrated module frames with respect to the base frame can also be computed as

$$T^c_{0,1}(0) = \begin{bmatrix} .0400 & .9988 & -.0268 & -.119 \\ -.0196 & .0276 & .9994 & 2.018 \\ .9990 & -.0394 & .0207 & 349.061 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

$$T^c_{0,2}(0) = \begin{bmatrix} .9976 & -.0669 & -.0196 & -8.548 \\ .0209 & .0190 & .9996 & 315.370 \\ -.0665 & -.9976 & .0204 & 354.605 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Therefore, the actual twists expressed in the calibrated module frames, denoted by $s^a_i[c]$ (i=1, 2), can be given by:

$s^a_1[c]$= $Ad^{-1}_{T^c_{0,1}(0)} Ad_{T^a_{0,1}(0)} s^a_1$=$(0,0,0,1,0,0)^T$=$s_1$; and

$s_2^a[c]= Ad^{-1}_{T^c_{0,2}(0)} Ad_{T^a_{0,2}(0)} s_2^a = (0,0,0,0,0,1)^T = s_2.$

The results also show that the actual twists expressed in the calibrated module frames retain their nominal coordinates so that our initial assumptions are valid.

6.3.2 Calibration of a SCARA Type Robot

A 4-DOF (RPFRFR) SCARA type modular robot is shown in Fig. 6.7. This robot consists of seven serially connected modules of different types, i.e., L_{r1}, L_{p1}, L_{c1}, L_{r1}, L_{c2}, L_{r2}, and L_{c2}. The last two revolute joint (J_{r2} and J_{r3}) axes are normally parallel with each other. The AIM is given as follows.

$$\mathcal{A}(\tilde{G}) = \begin{bmatrix} (z,-y) & 0 & 0 & 0 & 0 & 0 & L_{r1} \\ (-z,y) & (z,y) & 0 & 0 & 0 & 0 & L_{p1} \\ 0 & (x,-y) & (z,y) & 0 & 0 & 0 & L_{c1} \\ 0 & 0 & (-x,-y) & (z,y) & 0 & 0 & L_{r1} \\ 0 & 0 & 0 & (-x,-y) & (z,y) & 0 & L_{c2} \\ 0 & 0 & 0 & 0 & (y,-x) & (z,x) & L_{r2} \\ 0 & 0 & 0 & 0 & 0 & (z,x) & L_{c2} \\ J_{r1} & J_{p1} & J_{f1} & J_{r2} & J_{f3} & J_{r3} & 0 \end{bmatrix}.$$

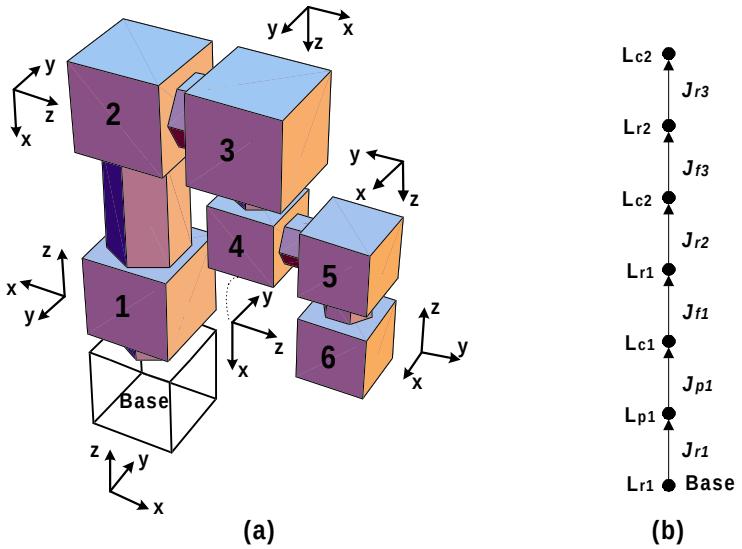


Figure 6.7: A SCARA type modular robot configuration

According to the given AIM, the calibration model can be automatically generated. Similar to the previous example, we first preset small geometric errors dp_i, ds_i, and dq_i

($i = 1, 2, \ldots 6$) into Dyad i (Table 6.4), and then compute a number of measured end link poses ($m = 40$). For convenience, the dp_i in each of the dyads is preset to the identical value, i.e., dp_i=(2.,2.,2., .02., .02, .02)^T. Since both joint 3 and 6 are fixed joints, ds_3, ds_5, dq_3, and dq_5 are all set to zero. In order to demonstrate the robustness of the proposed algorithm, an uniform random measurement noise dp'=$(dx', dy', dz', \delta x', \delta y', \delta z')^T$ is also added into the measured end link poses, where dx', dy', and $dz' \in$ U[-0.1mm,0.1mm], and $\delta x'$, $\delta y'$, and $\delta z' \in$ U[-0.001rad, 0.001rad]. The calibration results are shown in Table 6.5 and 6.6 and Fig. 6.8.

Table 6.4: Preset geometric errors for the SCARA robot

	Δ_i	ds_i	dq_i
Dyad 1	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 2., 0, 0.02, 0, -0.0002)^T$	0.02
Dyad 2	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0.0002, 0.02, 0, 0, 0, 0)^T$	0.02
Dyad 3	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 0, 0, 0, 0, 0)^T$	0
Dyad 4	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 0, 2., -0.0002, 0.02, 0)^T$	0.02
Dyad 5	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 0, 0, 0, 0, 0)^T$	0
Dyad 6	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(2., 0, 0, 0, 0.02, 0.0002)^T$	0.02

Table 6.5: Identified kinematic errors for the SCARA robot

	dp_i
Dyad 1	$(-0.110, 2.016, -0.661, 0.0171, 0.0413, -0.1044)^T$
Dyad 2	$(0.628, -0.020, 11.561, -0.1634, 0.0405, -0.0024)^T$
Dyad 3	$(0, 0, 0, 0, 0, 0)^T$
Dyad 4	$(0.675, 0.001, 11.581, 0.1447, 0.0010, 0.0800)^T$
Dyad 5	$(0, 0, 0, 0, 0, 0)^T$
Dyad 6	$(0.001, 3.577, -0.678, 0.0566, -0.0192, 0.0618)^T$
Compensation Dyad	$(0.001, -2.010, 0.741, 0.0201, -0.0005, 0.0338)^T$

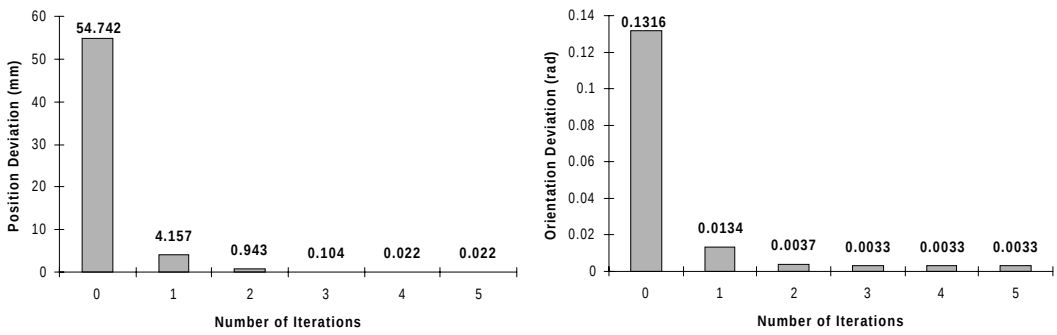


Figure 6.8: Calibration convergence (SCARA robot)

Table 6.6: End link poses before and after calibration (SCARA robot)

Joint Angle	q=(1.3259,.0174,0,.7298,0, 2.2093)^T	q=(-.2129,32.721,0,1.4841, 0, 1.9221)^T
Nominal Pose	$\begin{bmatrix} -.9991 & -.0425 & 0. & 312.442 \\ -.0425 & -.9991 & 0. & 493.938 \\ 0. & 0. & 1. & 387.517 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -.0516 & .9987 & 0. & -423.268 \\ -.9987 & -.0516 & 0. & 188.793 \\ 0. & 0. & 1. & 420.221 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Measured Pose	$\begin{bmatrix} -.9946 & -.1019 & -.0180 & 286.698 \\ .1018 & -.9948 & .0053 & 479.362 \\ -.0184 & .0034 & .9998 & 349.532 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -.1092 & .9907 & -.0808 & -389.917 \\ -.9939 & -.1077 & .0222 & 155.561 \\ .0133 & .0827 & .9965 & 363.005 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Calibrated Pose	$\begin{bmatrix} -.9947 & -.1020 & -.0151 & 286.686 \\ .1019 & -.9948 & .0068 & 479.356 \\ -.0157 & .0052 & .9999 & 349.509 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -.1094 & .9907 & -.0813 & -389.917 \\ -.9939 & -.1077 & .0254 & 155.555 \\ .0164 & .0836 & .9964 & 363.005 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Since joint 3 in Dyad 3 (modules 2 and 3) and joint 5 in Dyad 5 (modules 4 and 5) are fixed joints, the two modules in both Dyad 3 and Dyad 5 are rigidly connected. For simplicity, we consider each of these dyads as a rigid body. By doing so, the kinematic error in such a dyad will be lumped into its preceding dyad and succeeding dyad. For instance, the preset error dp_3 in Dyad 3 will be lumped into the kinematic errors of both Dyad 2 (modules 1 and 2) and Dyad 4 (modules 3 and 4). Thus, instead of identifying dp_3 and dp_5, we assume $dp_3 = dp_5 = (0, 0, 0, 0, 0, 0)^T$.

The computation results also show that the average positioning accuracy is significantly increased after a few iterations regardless of the existence of the measurement noise. This robot contains various types of joints, i.e., prismatic joint (joint 2), fixed joints (joints 3 and 5), and revolute joints (joints 1, 4, and 6) in which two adjacent ones (joints 4 and 6) are close to parallel. Hence, the proposed algorithm is a general and singular-free approach for the calibration of serial robots.

6.3.3 Calibration of a Tree-structured Robot

In this example, we wish to calibrate a tree-structured modular robot as shown in Fig. 6.4. According to the given AIM and path matrix, the calibration model can also be automatically generated. Similar to previous examples, we first preset small geometric errors dp_i, ds_i, and dq_i ($i = 1, 2, \ldots 8$) into Dyad i (Table 6.7), and then compute a number of measured poses (m=40) for each of the two end link frames. All dp_i in the dyads

are still preset to the identical value, i.e., dp_i=(2.,2.,2., .02., .02, .02)^T. An uniform random measurement noise is also added into the two measured end link poses, which is the same as the previous example. The calibration results are shown in Table 6.8 and 6.9 and Fig. 6.9 and 6.10. Note that for Table 6.9, the corresponding joint angles q = (1.5403,1.0310,0,0.3315,-0.6231,82.032,-2.5323,1.3762)^T.

Table 6.7: Preset geometric errors for the tree-structured robot

	dp_i	ds_i	dq_i
Dyad 1	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 0, 2., 0.02, 0.0002, 0)^T$	0.02
Dyad 2	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 0, 2., 0.02, -0.0002, 0)$	0.02
Dyad 3	$(0, 0, 0, 0, 0, 0)^T$	$(0, 0, 0, 0, 0, 0)^T$	0
Dyad 4	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 2., 0, 0.0002, 0, 0.02)^T$	0.02
Dyad 5	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 0, 2., 0, 0.02, 0.0002, 0)^T$	0.02
Dyad 6	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 0.0002, 0.02, 0, 0, 0)^T$	2.
Dyad 7	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 0, 2., 0.02, 0.0002, 0)^T$	0.02
Dyad 8	$(2., 2., 2., 0.02, 0.02, 0.02)^T$	$(0, 0, 2., 0.02, 0.0002, 0)^T$	0.02

Table 6.8: Identified kinematic errors for the tree-structured robot

	dp_i
Dyad 1	$(0.078, -0.044, 1.998, 0.0201, 0.0172, 0.0398)^T$
Dyad 2	$(0.574, 1.087, 0.050, 0.0196, 0.0062, 0.0172)^T$
Dyad 3	$(0, 0, 0, 0, 0, 0)^T$
Dyad 4	$(-1.850, 0, 1.084, -0.0249, 0.0400, 0.0343)^T$
Dyad 5	$(7.695, 0.005, 0.572, 0.0590, -0.0306, 0.0605)^T$
Dyad 6	$(-1.852, -0.971, 0, 0.0045, 0.0111, -0.0001)^T$
Dyad 7	$(0, 1.961, 0.003, 0.0206, 0.0277, 0.0467)^T$
Dyad 8	$(0, 0.960, 1.963, 0.0228, -0.0388, 0.0370)^T$
Compensation Dyad (branch 1)	$(1.851, 0.970, -0.004, -0.0197, -0.0088, -0.0002)^T$
Compensation Dyad (branch 2)	$(-1.993, -0.994, -0.005, 0.0016, -0.0586, 0.0204)^T$

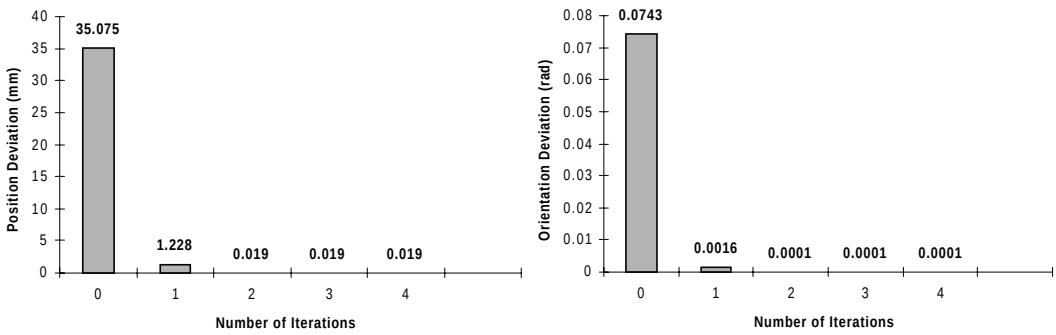


Figure 6.9: Calibration convergence for branch one

Table 6.9: End link poses before and after calibration (tree-structured robot)

	Branch 1 (Module 6)	Branch 2 (Module 8)
Nominal Pose	$\begin{bmatrix} -.8574 & .1383 & -.4957 & 168.263 \\ .0262 & -.9502 & -.3105 & 834.453 \\ -.5140 & -.2792 & .8111 & 646.864 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -.8753 & -.3480 & .3357 & 304.480 \\ .4707 & -.4543 & .7563 & -354.641 \\ -.1107 & .8200 & .5615 & 399.491 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Measured Pose	$\begin{bmatrix} -.8581 & .1666 & -.4858 & 156.437 \\ .0227 & -.9327 & -.3600 & 814.671 \\ -.5130 & -.3199 & .7965 & 699.125 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -.8892 & -.3799 & .2549 & 295.020 \\ .4299 & -.5032 & .7497 & -343.445 \\ -.1566 & .7762 & .6108 & 406.097 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Calibration Poses	$\begin{bmatrix} -.8581 & .1667 & -.4857 & 156.451 \\ .0226 & -.9327 & -.3600 & 814.665 \\ -.5130 & -.3199 & .7965 & 699.127 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -.8892 & -.3786 & .2566 & 294.994 \\ .4298 & -.4995 & .7522 & -343.445 \\ -.1566 & .7792 & .6069 & 406.088 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

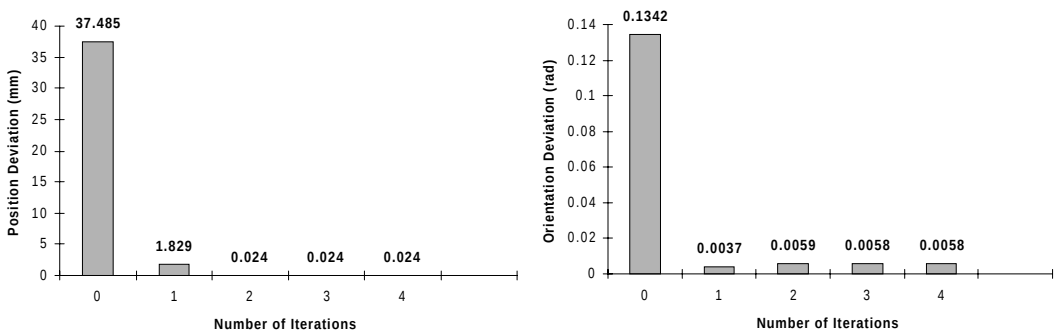


Figure 6.10: Calibration convergence for branch two

The computation results also shows that the average positioning accuracy in each of the two end links is significantly increased after a few iterations regardless of the existence of the measurement noise. This robot has a tree-structured topology and contains various types of joints, i.e., the prismatic joint, fixed joints, and revolute joints. The proposed algorithm therefore is not only suitable for serial robots but also suitable for tree-structured robots.

6.4 Discussion

Based on the local POE formula, a set of compact kinematic calibration models have been formulated for modular reconfigurable robots. The kinematic errors are attributed to the initial poses in the dyads so that the calibration models are significantly simplified. To a certain extent, the kinematic calibration is a process that redefines a set of new module frames on the modules, in which the kinematics under its description reflects the robot's actual characteristics. A least-squares algorithm is also employed to solve the calibration result iteratively. Three simulation examples for calibrating a 3-module robot (2-DOF), a 7-module SCARA type (4-DOF) modular robot, and a 9-module (7-DOF) tree-structured modular robot are demonstrated. The results have shown that the calibration models are simple, effective, and robust for modular robot applications without suffering from singularity problems. These algorithms are configuration-independent and can be applied to any tree-structured modular robot regardless of the number of DOFs and types of joints. The overall calibration algorithm can also be applied to industrial robots and other modular robot system without any modification. Implementations of the proposed algorithm on an ASEA IRB6/2 robot (5-DOF) and a tree-structured modular robot constructed by MoRSE modules (from AMTECH GmbH of Germany) have been completed and are successful. The experimental results conducted on these two real robots show the improvement of positioning accuracy to the magnitude in the order of the robots' repeatability. The detailed experimental calibration results can be found in [122, 123].

Chapter 7

Optimization of Modular Robot Configurations

This chapter discusses issues pertinent to the optimization of modular robot configurations. Unlike the existing industrial robots which are designed to perform general tasks, a modular robot system can provide optimal robot configurations to be suited for specific task requirements through proper selection and reconfiguration of modules. Hence, how to map the task specifications into the robot configuration design becomes an imperative subject for a modular robot system.

The modular design methodology also implies that one can build a robot configuration with fewer numbers of modules and DOFs to perform a task that is usually done by a universal industrial robot with 5 or 6 DOFs. With fewer numbers of modules and DOFs, a modular robot would have a simple configuration and a low power consumption rate, so that it can perform the task effectively. The Minimized Degrees of Freedom (MDOF) concept is therefore introduced in optimizing modular robot configurations [23, 138, 139]. The objective function is defined as the weighted sum of the numbers and types of modules in a robot, which is obviously to be minimized. To ensure the feasibility of the design result, several kinematic performance measures are considered as design constraints. A set of assembly rules is embodied in the search algorithm to eliminate all mechanically infeasible configurations from the search space. Based on the classical genetic algorithms (GA) and effective coding schemes defined on AIMs, an evolutionary algorithm (EA) approach is employed to search for optimal solutions. A virtual module

concept is introduced into the coding schemes to guarantee the operation of EA. Another benefit from the virtual module concept is that both type and dimension synthesis can be performed simultaneously in a uniform way. The goal of this chapter is to develop an algorithm to determine an MDOF robot configuration for a given task from a large pool of possible robot configurations. Here we focus only on the fixed base, serial type modular robots without end-effectors.

This chapter is organized as follows. Section 7.1 reviews the recent research efforts pertaining to the optimal design of robot configurations. Section 7.2 gives an introduction about the general design methodology. The optimization model dealing with the issues of the task specification, the design parameters, the searching space, the objective function, and performance constraints is discussed in Section 7.3. In Section 7.4, an evolutionary programming approach, based on the classical genetic algorithms and effective coding schemes, is addressed. Examples that demonstrate the problem solving strategy are presented in Section 7.5.

7.1 Introduction

There has been an increasing interest in studying the kinematic properties of robots recently. Most of these efforts are focused on identifying and quantifying certain qualitative features of the robot performance - performance measures. These measures are then used to evaluate and optimize robot configurations. Among these performance measures, the workspace volume and manipulability (dexterity) have been studied extensively in the literatures.

The workspace of a robot is defined as the set of all end-effector poses (described by the end-effector frame) which can be reached by some choice of joint angles. The relationship between a robot's kinematic parameters and its workspace is first addressed in the early work of Roth [111]. Now it is often customary to distinguish the workspace into the *reachable* workspace and the *dexterous* workspace [71]. The latter is the set that the end-effector frame can reach with arbitrary orientations, while the former is the set that the end-effector frame can reach with at least one orientation. Obviously, the dex-

terous workspace is a subspace of the reachable space. Both analytical and numerical methods have been used for characterizing these workspaces. Among the analytical approaches, a topological analysis of robot workspaces is given by Gupta and Roth [50], and Gupta [47]. In their works, the shapes of reachable workspace and dexterous workspace are analyzed. The concepts of workspace holes and voids are defined, and their existence conditions are identified. Further analytical studies of workspaces are addressed in Freudenstein and Primrose [41], where the relationships between robot kinematic parameters and workspace parameters are developed. This work is followed by an application to workspace optimization for 3-DOF robots [80]. A more general analysis of workspace optimization is presented by Paden and Sastry [92]. The optimal criteria of a serial type robot are defined in terms of the work-volume subject to a constraint on its length, and the well-connected workspace - the ability to reach all positions in its workspace in every pose. It is shown that the optimal design of a 6-R robot is an *elbow manipulator* (in which (1) the fourth, fifth, and sixth joint axes intersect at one point; (2) the Denavit-Hartenberg twist angles are 0 or 90 degrees; and (3) either the link length offset or the joint offset is zero for each link. A geometric approach is given by Park [101], in which the workspace volume is precisely defined by regarding the rigid body motion as a *Riemannian manifold*. The significance of this work is that the so defined workspace volume depends only on the joint axes, while the lengths of links and the fixed reference frames need not to be taken into account. Numerical studies of the robot workspace have been carried out by Kumar and Waldron [71], Yang and Lee [134], Tsai and Soni [125], and Yang and Hang [135]. These studies rely on numerical methods to generate projections of the workspace. The advantage of the numerical schemes over the analytical ones is that kinematic constraints can be easily included. However, the more general design principles and intuitions are more difficult to obtain, and the computation for the description of the workspace becomes very time consuming.

The manipulability or dexterity of a robot can be described as the ability to move and apply force and torque in arbitrary directions. One of the early papers to consider manipulability is the work of Salisbury and Craig [114]. The condition number of the Jacobian matrix (J), $\mathrm{Cond}(J) = \sigma_{max}/\sigma_{min}$, is proposed to measure the transmission of the joint

velocity errors to the fingertip velocity errors, where σ_{max} and σ_{min} are the maximum and minimum singular value of the Jacobian matrix respectively. A clearer description of robot's manipulability is given by Yoshikawa [141], where the *manipulability ellipsoid* is introduced. The product of all singular values of the Jacobian matrix, which is equal to $\sqrt{\det JJ^T}$, is defined as the manipulability measure. This scalar quantity can measure how much the end-effector moves for a given infinitesimal movement of joint angles, averaged over all directions. In order to measure the closeness of a robot posture to a singularity, the *condition index*, $CI(J) = 1/\mathrm{Cond}(J) = \sigma_{min}/\sigma_{max}$, has been proposed by Angeles [5]. Definitely, the range of CI is from 0 to 1. CI becomes zero if the robot is in a singular posture. A robot is called isotropic if its CI attains one. All the manipulability or dexterity measures mentioned above are local indices in the sense that they characterize the manipulability or dexterity of a robot at a given posture. They are useful for evaluating the motion ability of an existing robot. For the design of a general-purpose robot, however, global ones may be more desirable. A straightforward way that derives global dexterity measures is to integrate the local ones over some region in the configuration space [46]. A geometric approach is given by Park [101], in which a global description of dexterity is obtained by introducing the harmonic mapping technique.

Besides the workspace volume and manipulability (or dexterity) measures, several other performance indices are also defined by different researchers such as the joint range availability [70], the solvability of the kinematic equations, the mechanical constructability [58], the accuracy achievability [27], and other nonlinearity and redundancy measures [36].

Most of the performance measures mentioned above are proposed for analysis and design of general-purpose robots. For the design of a modular robot that is only aimed at performing a given task, the effectiveness of these performance measures remains unclear and needs to be studied. For example, the workspace volume may be a main performance index for a general-purpose robot. It becomes less important for evaluating a modular robot that is designed to cater for the specific task (if the task can be well executed).

Past researches on the optimal design of modular robots is mainly focused on the so-called task-based design (TBD) problem. The goal of TBD is to design a robot that is optimally suited to perform a given task. The studies on TBD are introduced by Paredis [95] for

the *Reconfigurable Modular Manipulator System* (RMMS) [115]. In this work, the TBD problem is divided into several stages i.e., kinematic design, dynamic design, controller design, and path planning. A robot task is defined as a set of working poses (positions and orientations) in the task space. The simulated annealing algorithm is employed to search for the optimal solution. The objective function is a penalty function in which penalties for violating task constraints, such as reachability and obstacle constraints, are combined in a weighted sum fashion.

A more comprehensive study on the design of task specific fault tolerant manipulator is also given by Paredis [93]. Such a TBD problem is also formulated as design optimization problem. The task description consists of a timed Cartesian path to be followed by the end-effector, and obstacle models in the manipulator workspace. An integrated evaluation function is proposed to deal with the tight coupling between the performance measures including joint angle, velocity, and torque limits, reachability, singularity avoidance, collisions with obstacles, and self-collision caused by the fault tolerance requirement. The objective criterion is defined as a desired trajectory with minimum energy consumption. Since many performance measures are considered, the evaluation procedure is very time consuming. To make this integrated approach computationally feasible, two techniques are addressed i.e., to include problem specific design knowledge in the search, and implement the search in an easily parallelizable agent-based paradigm on the basis of genetic algorithms. Even so, the computational cost is still expensive. As mentioned in [98], to derive a feasible solution (with 75 percent probability) for satellite docking task needs about 6 hours of CPU time on twenty-four Sparc workstations.

Kim and Khosla [68, 69] also focus on the TBD problem for the RMMS. They pose the kinematic design of modular robots as an optimization problem. Similarly, the task requirement is also defined as a set of working poses in the task space. A multi-population genetic algorithm (MPGA) is employed as the optimization algorithm. The idea is to have one GA per task pose to search for locally optimal designs, and then progressively increases the coupling between the individual GAs to arrive at one single globally optimal design for all the task poses. The design variables include the number of DOFs, Denavit-Hartenberg (D-H) parameters, and the base position. The dexterity measure is chosen

as the objective criterion. Several kinematic constraints, such as dimension constraints, joint constraints, and task specification constraints, are considered.

Chen [14, 15] formulates the task-oriented optimal configuration problem as a combinatorial optimization problem due to the discrete feature of modules. The formulation is based on an Assembly Incidence Matrix (AIM) representation of a modular robot. The task requirement is defined as a set of sequential working points (positions) in the task space. A task dependent objective function, termed the Assembly Configuration Evaluation Function (ACEF), is proposed. This objective function contains both task evaluation and structure evaluation criteria. An evolutionary algorithm based on genetic operators defined on AIMs is employed to solve this optimization problem [15].

The task optimization problem is also addressed by Ambrose [1] for the modular robot system built in University of Texas at Austin. The purpose of this work is to allow a user to specify the robot performance requirements and then to select the best of the available module combinations for a set of goals. Ten criteria are considered in order to evaluate the performance of a robot, i.e., mobility, payload, speed, workspace, weight, backlash, friction, stiffness, inertia, and bandwidth. Several optimization algorithms such as the exhaustive search, the sequential filters, the modified golden section, the steepest descent, the conjugate direction, and Taguchi methods are employed for the design of one and two-dimensional robots for the comparison purpose, but no explicit objective function is defined.

Besides the task requirements, most of previous works on TBD problem still emphasize on robot performance measures such as manipulability, singularity avoidance, and fault tolerance, thus tending to select a sophisticated robot configuration with more DOFs as the optimal one. The reason behind this is that a robot with more DOFs can easily satisfy the task requirements and performance measures. An optimal robot under such considerations is usually a redundant one, e.g., more than 6 DOFs for a 3-dimensional manipulator and more than 3 DOFs for a regional (positioning) robot arm. For most industrial applications, however, the operation speed is the main concern. A modular robot with simple kinematic structure - few modules and DOFs, is much more preferred in order to perform a given task swiftly. Therefore, how to design a MDOF modular robot

for a given task is crucial to a modular robot system.

7.2 General Design Methodology

As mentioned in [93], the optimal design of a modular robot configuration for a given task is a very difficult problem. It involves both type (topological structure) synthesis and dimension synthesis. The design space is discrete and grows exponentially with the number of modules in the inventory. Various performance measures are highly coupled and nonlinear, and to evaluate whether the constraints are satisfied and to which extent the optimality criteria are achieved are very computationally expensive.

To reduce the complexity of TBD, we divide the design problem into several stages as shown in Fig. 7.1. The *kinematic design* is the first stage, i.e., Stage (1), because of its importance. It determines the kinematic configuration of the robot based on the kinematic requirements of a given task. The output of this stage is one "optimal" and several "sub-optimal" robot configurations. Since we aim at designing simple robots with minimized DOFs, it is more difficult to perform trajectory planning in the task space (Cartesian space). Instead, a joint space trajectory planning algorithm is to be employed for the derived robot configurations in Stage (2). The joint angle, velocity, and acceleration limitations thus can be easily considered in this stage. After that, a 3-D graphic simulation, based on the derived joint trajectory, will be given for the visualization of task execution and collision detection in Stage (3). Then, the computation of inverse dynamics will be performed for the investigation of joint torque limitations and the controller design in Stage (4). If the "optimal" robot configuration can successfully pass Stages (2), (3) and (4), it is definitely accepted as the final design; otherwise, the best one among the "sub-optimal" robot configurations will be selected and cast into the last three stages. This procedure will repeat until a survived one is obtained. If all the "sub-optimal" ones still fail to pass Stages (2), (3) and (4), the whole design procedure should be restarted and repeated until a desired robot configuration is derived. If it is the case that an acceptable solution cannot be found after a number of redesigns, it may imply that the given task is unreasonable and needs to be modified.

Note that the module specifications listed in Chapter 2 provide the kinematic and dynamic parameters of modules. For the purpose of collision detection, a 3-D graphic model is also to be created for each module.

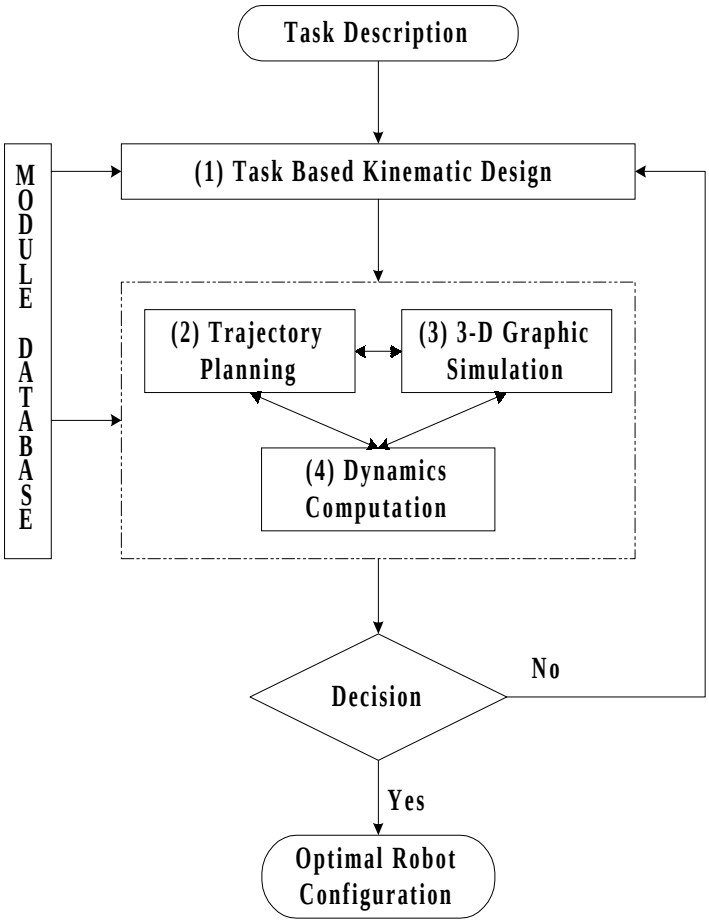


Figure 7.1: Flow chart of the iterative design methodology

To a certain extent, this design methodology is an iterative method. Iterations may occur to the whole design procedure or within several stages. It is true that such a method cannot guarantee that a desired robot configuration can be obtained within a few iterations. Compared to an integrated design method [93], however, this method can avoid spending unnecessary computation time to evaluate kinematically illegal individuals. This design methodology is based on the consideration that an optimal robot configuration should be kinematically optimal or sub-optimal one because the kinematic performance of a robot is critical for the task realization. Hence, in most cases, the proposed method is more efficient and practical. In the following sections of this chapter, we will mainly focus on the optimal kinematic design of modular robot for it is the most important part

of the whole design procedure.

7.3 Optimization Model

The primary issue in the optimal kinematic design of a modular robot is to formulate a task-oriented optimization model. This model consists of three main parts: design parameters, the objective function, and constraints.

7.3.1 Definition of Robot Tasks

There are many different levels of details at which a robot task can be defined [93]. As an example, consider the assembly of a printed circuit board (PCB). At the highest level, this task could simply be described as: "assemble a PCB". At an intermediate level, the task will be decomposed into many subtasks such as "locate a chip in the parts feeder", "pick up the chip", and then "insert the chip into the board", etc. At the lowest level, all the details are included, such as "move the end-effector from point A to point B", "exert a force of n Newtons on the end-effector in x direction to grasp a chip at point B", and "move the end-effector from point B to point C", etc. If the robot task is described at higher levels, a task planner is needed to convert the higher level task descriptions into the low level task descriptions. Although the task planning problem is also a challenging issue, it is not the main concern of this thesis. Here, we would like to adopt a low level task description such that the robot motion sequence and trajectory are provided for task execution.

The execution level of the robot motion sequence and trajectory are adopted to define a robot task for two reasons. First, with a prescribed robot motion sequence and trajectory, one can immediately determine the capability of a modular robot configuration to carry out the task. Furthermore, if the robot is unable to follow the motion, it is unnecessary to proceed any further with the evaluation procedure. Secondly, it is very difficult to define a measure to quantify the "goodness" of a robot while performing a task with a very vague definition [14].

For simplicity, a robot task is specifically defined as a collection of working points or poses, $\mathbf{w}_p$, to be followed by the end-effector or the end link in the task space [14], where $\mathbf{w}_p \in \Re^{3\times 1}$ or $\mathbf{w}_p \in SE(3)$. If the robot is to follow a prescribed path, this task can be approximated by a set of points or poses along the path. Other task specifications such as obstacle avoidance, position accuracy, and static capability at task points or poses can also be included as part of the definition of a task. Note that in the following context, the term "point" refers to the position of the end link frame and "pose" refers to both position and orientation of the end link frame.

7.3.2 Design Parameters and the Search Space

The parameters that determine a modular robot configuration are: (1) the number of the constituting link modules - N, (2)the link module types - L ($L \in \{L_{ci}, L_{ri}, L_{pi}\}$, $i = 1, 2$), and (3) the relative assembly orientations (port vectors) of the consecutive modules - P. When these parameters are all specified, a robot configuration is uniquely defined. Hence, we consider the set of design parameters X as:

$$X \rightarrow \{N, L, P\}. \tag{7.1}$$

To fully employ the power of discrete optimization technique in the MDOF problem, we introduce a virtual link module type and a virtual joint type (connecting to the output socket of the virtual link module), denoted by L_v and J_v respectively, into the inventory of modules. The virtual links and joints function as "slack parameters" to compensate for the fixed size representation of the candidate solutions for certain search/optimization techniques like genetic algorithms. Both L_v and J_v do not have physical dimensions and do not exist in the actual robot assembly. We assume that the virtual link modules (L_v) exist in the abstract kinematic description of the robot and have the virtual appearance of cubic link modules. To illustrate this virtual module concept, Fig.7.2 shows the kinematic graph representation of three different modular robots. Physically they have different numbers of modules, i.e., five, four, and three modules respectively, but the dimensions of their kinematic graphs as well as AIMs remain identical due to the introduction of the virtual modules.

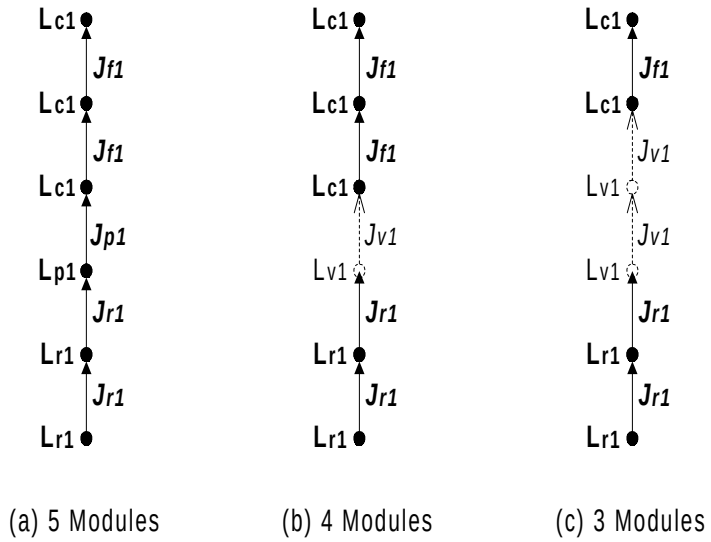


Figure 7.2: Virtual module in the kinematic structure

According to the definition of AIM, all the design parameters N, L, and P can be clearly indicated in an AIM. For every possible design (N,L,P), there exists a unique AIM. Therefore, the design parameters can be defined in AIM format. The size of the search space is equal to the number of all possible AIMs. Since the joints (connectors) depend on the links they are connected to, they are not independent parameters and are not considered as design parameters. Therefore, the last row of the AIM can be eliminated.

Excluding the virtual module, six kinds of modules ($L \in \{L_{ci}, L_{ri}, L_{pi}\}$, i=1, 2) can be chosen for robot assembly. Each module has six connecting sockets, and a module can be connected in four different orientations relative to its preceding module even if the connecting sockets remain unchanged. The search space thus becomes extremely large. For example, an assembly of two connected modules will have a total of $6 \times 6 \times 4 \times 6 \times 6 = 5184$ possible configurations. Likewise, consider an n-module serial type robot without any virtual module. The number of all possible configurations, N_{pc}, can be given by

$$N_{pc} = 4^{(n-1)} \times 5^{(n-2)} \times 6^{2n}. \tag{7.2}$$

Equation (7.2) shows that the search space grows exponentially with the number of modules in the robot. If the maximum allowed number of modules in a robot is equal to m and suppose that a robot consists of at least two real modules, such that the number of virtual modules may be $0, 1 \ldots,$ or $m-2$, then the total number of possible configurations

can be calculated by

$$N_{total} = \sum_{i=2}^{m} 4^{(i-1)} \times 5^{(i-2)} \times 6^{2i}. \tag{7.3}$$

Taking $m = 6$ as an example, the total number of the possible assembly configurations is 1.4×10^{15}. Note that in these configurations, there are some kinematically identical ones which are not eliminated from Equation (7.2) and (7.3). In [14], Chen proposes an algorithm that can enumerate all of the distinct modular robot configurations from a given set of modules. Chen's enumeration algorithm can effectively reduce the size of the search space so that it is more suitable for the case of exhaustive search. Nevertheless, to obtain an optimal robot configuration from such a large and discrete search space is definitely time consuming.

According to the module designs, when we assemble a complete robot with different sizes of modules, a general rule is that the large modules should precede the small modules which are usually located at the distal ends. In the search space, however, there still exist such robot configurations that: (1) small size modules are connected between two large size modules; (2) large size modules are connected between two small size modules. From mechanical point of view, a robot with such assembly patterns is an unreasonable structure because of the higher energy consumption and lower loading capability. To avoid this problem, we propose here a two-step configuration design method: the *optimization* step and *module replacement* step.

In the first step, only the large size modules are considered in the inventory of modules, i.e., $L \in \{L_{c1}, L_{r1}, L_{p1}, L_v\}$, so that many mechanically infeasible configurations can be avoided and the search space will be significantly reduced. The total number of the possible assembly configurations can be computed by

$$N_{total} = \sum_{i=2}^{m} 4^{(i-1)} \times 5^{(i-2)} \times 6^{i} \times 3^{i}. \tag{7.4}$$

Also taking $m = 6$ as an example, the possible assembly configurations are reduced to 2.2×10^{13} (about one sixtieth of the size of the original search space).

In the second step, the large modules in the optimal (or sub-optimal) robot configuration will be replaced with small modules of the same types as many as possible. The module replacement will be done from the end link module to the base link module one by one.

Each module replacement will also satisfy the computation results of Stages (2), (3), and (4) in Fig. 7.1 and the assembly rules to be mentioned in Section 7.4. Moreover, when we replace a large module with a small one in a robot, the length of the corresponding connectors should be enlarged accordingly in order to remain the kinematic parameters unchanged. It follows that after the *module replacement* step, the kinematic structure of the robot still remain the same. The resulting robot configuration, on the other hand, becomes more reasonable for the given task. Since this chapter is focused on the kinematic design of modular robot configurations, only the first step, the *optimization* step, is addressed here.

7.3.3 The Objective Function

The objective function is to evaluate the "goodness" of a robot assembly configuration for a given task. Since the performance measures are defined to quantify the features of robot performance, a straightforward way to define the objective function is to list all necessary performance measures such as reachability, manipulability, and fault tolerance index, and then choose an important one among them as a single objective function [14, 15, 69] or combine some or all of them in a weighted sum as a multi-objective function [28, 95]. Performance measures that are not included in the objective function can be treated as constraints. However, if the objective function is only focused on robot performance measures, there will be a tendency of selecting a robot configuration with more DOFs as the optimal one. Moreover, since the type (topology) synthesis will also be performed in the optimization procedure, some commonly used performance measures become invalid when used as objective criteria. For example, the manipulability and workspace measures are both powerful criteria for evaluating robots with the same topological structure, but it is unfair when using them to evaluate robots with different topological structures.

Because of the modular design, a modular robot will have most of its actuators being located on the moving arm itself. Using excessive modules, especially the actuators, will certainly lower the robot's loading capability. Hence, to use a minimal number of modules as well as DOFs to construct a robot assembly for a task becomes our main concern. The objective function F is therefore defined as a weighted sum of the numbers of different

types of modules to be employed in a robot:

$$F = k_{p1}N_{p1} + k_{r1}N_{r1} + k_{c1}N_{c1} \tag{7.5}$$

where N_{p1}, N_{r1}, and N_{c1} represent the numbers of large size prismatic, revolute, and cubic link modules respectively; k_{p1}, k_{r1}, and k_{c1} are the user-defined weightages of N_{p1}, N_{r1}, and N_{c1} respectively. All these weightages are positive constants and reflect the relative contribution of their corresponding link modules to the objective function. To design MDOF robots, the value of objective function is to be minimized. In this case, the weightages in Equation (7.5) would actually act as penalty coefficients. When we wish to minimize the number of revolute and prismatic joints (DOFs), we will set k_{r1} and k_{p1} to be much larger than k_{c1}, as cubic link modules do not contribute to the DOFs of the robot. If we wish to have more revolute joints than prismatic joints (i.e., $N_{r1} > N_{p1}$), then the weightage k_{p1} is set to be greater than k_{r1}; if we treat both types of joints to be of equal importance, then k_{p1} is set to be equal to k_{r1}; otherwise, k_{r1} is set to be greater than k_{p1}. These basic rules are only for determining the relative importance of the types of modules. Suitable values of k_{p1}, k_{r1}, and k_{c1} have to be determined by trial computations.

Since a classical genetic algorithm (GA) maximizes the objective function during the search procedure, we define the reciprocal of $F(F > 0)$, $1/F$, as the fitness function for the search algorithm. Without loss of physical interpretation, the term "fitness" in the context still refers to the objective function value as given by Equation (7.5).

7.3.4 Performance Constraints

The performance constraints are defined to ensure the feasibility of a robot configuration while performing the given task. The constraints considered here are:

- Reachability;
- Joint Range Availability;
- Manipulability;
- Mechanical Constructability.

The **reachability constraint** is used to inspect whether all of the given task points/poses are within a robot's reachable workspace. If this constraint is satisfied, the robot configuration is able to perform the given task kinematically. To study the reachability constraint of a robot, we transform the problem into the inverse kinematics problem. Because a modular robot has unfixed assembly configurations, the numerical inverse kinematic algorithm presented in Chapter 4 is employed. For a candidate robot, if an inverse solution is found at each of the task points/poses, the reachability constraint is considered to be satisfied. Suppose that each of the given task points or poses is $w_{pi} \in \Re^{3\times 1}$ or $SE(3)$ $(i = 1, 2, \ldots, m)$, the reachability constraint for a robot with $n+1$ modules is given by:

$$T_{01}(0)e^{\widehat{s}_1 q_{1i}}T_{12}(0)e^{\widehat{s}_2 q_{2i}}\ldots T_{n-1,n}(0)e^{\widehat{s}_n q_{ni}} = w_{pi}. \tag{7.6}$$

The **joint range availability constraint** is defined to detect whether the joint angles derived at each task point/pose are within the joint angle limitations. The joint range availability constraint for a robot with $n+1$ modules is then given by

$$a_i \geq q_i \geq b_i (i = 1, 2, \ldots, n), \tag{7.7}$$

where b_i and a_i are the lower bound and upper bound of joint i's displacement respectively.

The **manipulability constraint** here is only used to detect whether a robot is at or nearby a singular posture when its end-effector reaches each of the task points or poses, and thus it is a local manipulability index. For convenience, the *condition index* CI is employed to formulate the manipulability constraint:

$$\sigma_{min}/\sigma_{max} \geq \epsilon, \tag{7.8}$$

where σ_{min} and σ_{max} are the minimum and maximum singular values of the robot Jacobian matrix respectively. ϵ is the user defined lower bound of CI for singularity avoidance. It usually takes a small value, 0.001 for instance.

The **mechanical constructability constraint** is proposed to check the mechanical feasibility of a robot assembly. For example, structures with two or more modules attached to the same connecting socket of an identical module are obviously infeasible. To avoid such structures, mechanical constructability constraints have to be given. However, if

there are excessive constraints, one will run the risk of creating a search algorithm that spends most of its time evaluating illegal individuals [86]. Instead of formulating these constraints, several assembly rules are built into the search algorithm so that mechanically infeasible structures can be intelligently avoided. The assembly rules for fixed base, serial type robots are:

1. A complete modular robot consists of at least two real modules.
2. The local coordinate (module) frame of the base module coincides with the world frame.
3. For each of the intermediate modules, only two out of the six connecting sockets will actually be used for assembly, i.e., the "input socket" which is connected with its proceeding module and the "output socket" which is connected with its succeeding module.
4. The base module only has a "output socket", and does not have any "input socket". On the contrary, the end module has a "input socket", and does not have any "output socket".
5. The moving socket of any joint module is always used as "output socket" such that the z direction of the module frame points to its succeeding module.

The first four assembly rules are focused on constructing a robot with a fixed base, serial type kinematic structure. The last rule is aimed at assembling a robot with a reasonable mechanical structure. Taking the fifth rule as an example, if the "output socket" of a revolute joint module is not the moving socket, it is functionally the same as a link module, but it is heavier and more expensive than the link module and is thus definitely unacceptable. Since the coding schemes for search algorithm are defined on AIMs, these assembly rules will inherently act on AIMs through an *AIM Generating Scheme* and genetic operators (mutation operators). As a result, the search algorithm will always evaluate mechanically feasible robot configurations.

7.4 Evolutionary Algorithm

Since both the design parameters and the search space are discrete in nature, combinatorial optimization techniques can be applied. Exhaustive search techniques can find the exact optimal solution, but the search space is so large that implementation of such an algorithm is inefficient. Random search techniques such as the genetic algorithm (GA) and the simulated annealing (SA) method are more suitable for such a problem [14, 97].

The evolutionary algorithm (EA) is a variation of GA with problem specific data representation and genetic operators [86]. It is a probabilistic search method based on the principle of evolution and hereditary of nature systems. In such an algorithm, a population of individuals for each generation is maintained. The individual is implemented with some data structure and is evaluated by a "fitness function" to give a measure of its "fitness". A new population is formed by selecting the more suitable individuals. In this selection procedure, an individual with larger "fitness" value ($1/F$) is more likely to be selected for the new generation. Some members of the new generation undergo transformations by the "genetic operators" to form new solutions. Probability rules are applied to determine the execution of the operators. Two types of genetic operators are considered in general: *crossover* type and *mutation* type. The crossover operation reorganizes data segments in several individuals to form new individuals (or the offspring). The mutation operator makes small changes in a single individual. After some number of generations, the individuals will converge to the optimal or nearly optimal solution.

7.4.1 Coding Scheme

It is reported that the computational efficiency and stability of GAs rely primarily on the data representation method, i.e., the coding scheme [86]. Classical GAs usually use fixed-length binary strings as a chromosome (the data structure) for its individuals and the genetic operators: the binary mutation and the binary crossover. Such a coding scheme when adopted for robot configuration designs becomes inconvenient for the binary string representation is not sufficient to reflect the nature of the design parameters. Also, a conversion scheme is required to transform the binary strings into the AIMs [14]. In

an evolutionary algorithm, on the other hand, chromosomes need not be represented by binary strings. The alteration process includes employing the problem-specific data structure to represent the chromosomes and modifying the "genetic" operators appropriately for the given data structure. Since the AIM representation conceptually defines the data structure of the solution space, we adopt the AIM as the chromosome and define the AIM related genetic operators. The main objective behind such an implementation is to move the evolutionary algorithm closer to the solution space.

7.4.2 AIM Generating Scheme

The AIM generating scheme is proposed to automatically generate the initial population of AIMs. This scheme consists of two parts: link type generation and port vector generation. Although both link types and port vectors are generated on the basis of the assembly rules, the generation procedure would still have a random feature. For example, based on the first and second assembly rules, we can still randomly choose either L_{c1}, L_{r1}, L_{p1}, or L_v as the base link. The input of this scheme is an inventory of link modules - $L = \{L_{c1}, L_{r1}, L_{p1}, L_v\}$ and the maximum allowable number of modules to be used in a robot - n_m. The output is an AIM satisfying all the assembly rules.

7.4.3 Genetic Operators on AIMs

Crossover Operator

The crossover operator applies to the rows of two AIMs. A cut-off row is randomly chosen. Then the rows in the two AIMs below the cut-off row (Fig. 7.3) are swapped to form two new AIMs. The probability p_c controls this operation.

Mutation Operators

The mutation operators act on non-zero entries of a single AIM. Since the non-zero entries in a modified AIM (without last row) can be divided into two categories: port vectors and link types (the last column), two types of mutations for both port vectors and link types are considered.

Before Swaping

$$A1 = \begin{bmatrix} (z,x) & 0 & 0 & 0 & Lp1 \\ (x,y) & (z,y) & 0 & 0 & Lc1 \\ 0 & (x,y) & (z,-y) & 0 & Lp1 \\ 0 & 0 & (-z,-y) & (z,-y) & Lr1 \\ 0 & 0 & 0 & (-z,x) & Lc1 \end{bmatrix} \quad A2 = \begin{bmatrix} (z,x) & 0 & 0 & 0 & Lr1 \\ (-x,z) & (z,-y) & 0 & 0 & Lr1 \\ 0 & (-x,-y) & (z,x) & 0 & Lc1 \\ 0 & 0 & (y,z) & (z,x) & Lp1 \\ 0 & 0 & 0 & (y,x) & Lc1 \end{bmatrix}$$

After Swaping

$$A1' = \begin{bmatrix} (z,x) & 0 & 0 & 0 & Lp1 \\ (x,y) & (z,y) & 0 & 0 & Lc1 \\ 0 & (x,y) & (z,-y) & 0 & Lp1 \\ 0 & 0 & (y,z) & (z,x) & Lp1 \\ 0 & 0 & 0 & (y,x) & Lc1 \end{bmatrix} \quad A2' = \begin{bmatrix} (z,x) & 0 & 0 & 0 & Lr1 \\ (-x,z) & (z,-y) & 0 & 0 & Lr1 \\ 0 & (-x,-y) & (z,x) & 0 & Lc1 \\ 0 & 0 & (-z,-y) & (z,-y) & Lr1 \\ 0 & 0 & 0 & (-z,y) & Lc1 \end{bmatrix}$$

Figure 7.3: Crossover operation

- **Link type mutation:** As shown in Fig. 7.4, the link type mutation acts on the last column of an AIM. The link type will be altered randomly, while the port vectors remain unchanged. The probability p_{ml} controls this operation. This operation will be subject to the assembly rules.

- **Port vector mutation:** As shown in Fig. 7.4, port vector mutation acts on the port vectors of an AIM. The port vector is altered randomly, while types of links remain unchanged. The probability p_{mp} controls this operation. This operation will also be subject to the assembly rules.

$$\begin{bmatrix} (z,x) & 0 & 0 & 0 & Lr1 \\ (-x,z) & (z,-y) & 0 & 0 & Lr1 \\ 0 & (-x,-y) & (z,x) & 0 & Lc1 \\ 0 & 0 & (-z,-y) & (z,-y) & \boxed{Lr1} \\ 0 & 0 & 0 & (-z,y) & Lc1 \end{bmatrix} \xrightarrow{\text{Link Type Mutation}} \begin{bmatrix} (z,x) & 0 & 0 & 0 & Lr1 \\ (-x,z) & (z,-y) & 0 & 0 & Lr1 \\ 0 & (-x,-y) & (z,x) & 0 & Lc1 \\ 0 & 0 & (-z,-y) & (z,-y) & \boxed{Lc1} \\ 0 & 0 & 0 & (-z,y) & Lc1 \end{bmatrix}$$

$$\xrightarrow{\text{Port Vector Mutation}} \begin{bmatrix} (z,x) & 0 & 0 & 0 & Lr1 \\ (-x,z) & (z,-y) & 0 & 0 & Lr1 \\ 0 & (-x,-y) & (z,x) & 0 & Lc1 \\ 0 & 0 & (-z,-y) & (z,-y) & Lr1 \\ 0 & 0 & 0 & (-z,x) & Lc1 \end{bmatrix}$$

Figure 7.4: Mutation operation

7.4.4 Implementation of the Evolutionary Algorithm

The proposed evolutionary algorithm is successfully implemented in *Mathematica*. As shown in Fig. 7.5, the input parameters include an inventory of link modules L, the population size n_p, the number of destination generation n_g, the maximum allowable number of modules n_m, the crossover operator p_c, and the mutation operators p_{ml} and p_{mp}. The first step is to randomly generate n_p AIMs for the initial generation by using the AIM generating scheme. It is followed by the task evolution procedure to compute the fitness value for each AIM (Fig. 7.6). After the task evolution procedure, a new generation of AIMs is produced through reproduction, crossover, and mutation operations. The entire process will be repeated until the predetermined number of generations n_g is reached. In the final generation, we choose the AIM with the smallest fitness value (F) as the optimal assembly configuration. Moreover, during the search procedure, the optimal one in each generation is also recorded. Several sub-optimal AIMs can thus be derived from the intermediate optimal configurations.

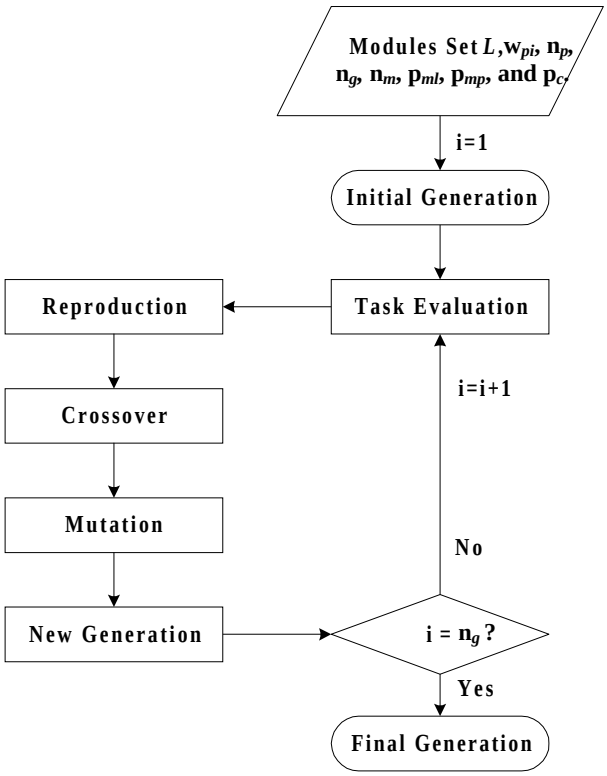


Figure 7.5: The kinematic design algorithm

The task evaluation procedure, as shown in Fig. 7.6, consists of two parts: performance

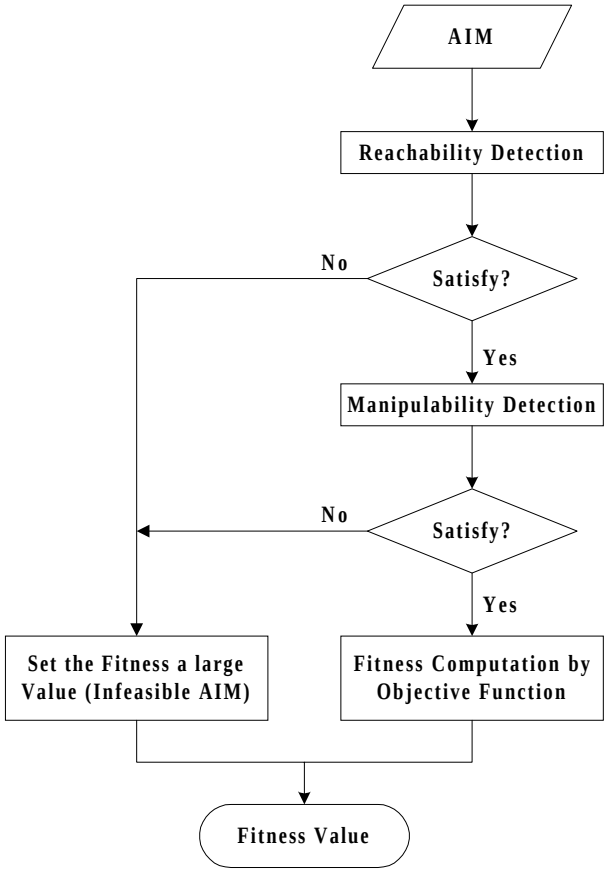


Figure 7.6: Task evaluation procedure

constraint detection and fitness value computation. If an AIM can pass all the constraints, i.e., reachability, joint range availability, and manipulability, its fitness can be calculated by Equation (7.5); otherwise, the fitness is assigned to an infinite large number to indicate that such an AIM is infeasible. In the real implementation, however, an infinitely large fitness value is not convenient for the reproduction (selection) procedure, so we replace it with a very large number to indicate that such an AIM consists of excessive modules and has little chance to be selected for the next generation.

Since the reachability constraint is very strict and is difficult to be satisfied, the evolutionary algorithm may become unstable. It may be the case that the final generation does not contain any acceptable solution even if the optimal individuals have ever been created in the intermediate generations. Hence, two techniques are employed to increase the stability of the algorithm.

- **Adaptive mutation operators:** the mutation operators (p_{mp} and p_{ml}) are dy-

namically changed according to the average fitness of each generation. The suitable relationship between mutation operators and the average fitness values is determined through trial computations. In general, the smaller the average fitness value, the smaller the mutation operators. If all the individuals in a generation are infeasible configurations, then the probability of mutation operations becomes very high. Consequently, this is equivalent to randomly selecting AIMs.

- **Eugenics:** the best AIM obtained previously will be retained from generation to generation until a better one is created. Then the new AIM will replace the old one.

By employing these two techniques, the algorithm becomes more stable and can always find acceptable solutions. However, if the given mutation operators and the population size are too small, one will run a risk that the algorithm may converge to a local minimum.

7.5 Computation Examples

In order to demonstrate the effectiveness of the proposed evolutionary algorithm, two examples are given in this section. In the first example, the given task is a set of working points (positions), termed the position design problem, while in the second example the given task is a set of working poses (positions/orientations), termed the pose design problem. In both examples, the user defined weightages for different types of link modules are taken as $k_{p1} = 1.5$, $k_{r1} = 1.0$, and $k_{l1} = 0.25$; the crossover operator is uniformly taken as $p_m = 0.25$. If an AIM cannot satisfy the constraints, its fitness is set to 20.

Example 7.1: Position Design Problem

In this example, we wish to design a fixed base serial robot that can pass through a set of task points listed in Table 7.1. We set $n_m = 6$ and $n_g = 50$. The adaptive probabilities for the mutation operators are listed in Table 7.2.

Table 7.1: Task points

Point 1	Point 2	Point 3	Point 4
(-212.52, -533.07, 0)	(247.48, -597.48, 0)	(641.18, -265.59,0)	(700.00,0,0)

Table 7.2: The adaptive probabilities for the mutation operators (position problem)

Average Fitness Value F_a	Link Mutation Operator	Port Mutation Operator
20	0.2	0.2
$10 \leq F_a < 20$	0.05	0.05
$5 \leq F_a < 10$	0.03	0.03
$3 \leq F_a < 5$	0.02	0.02
$F_a < 3$	0.015	0.015

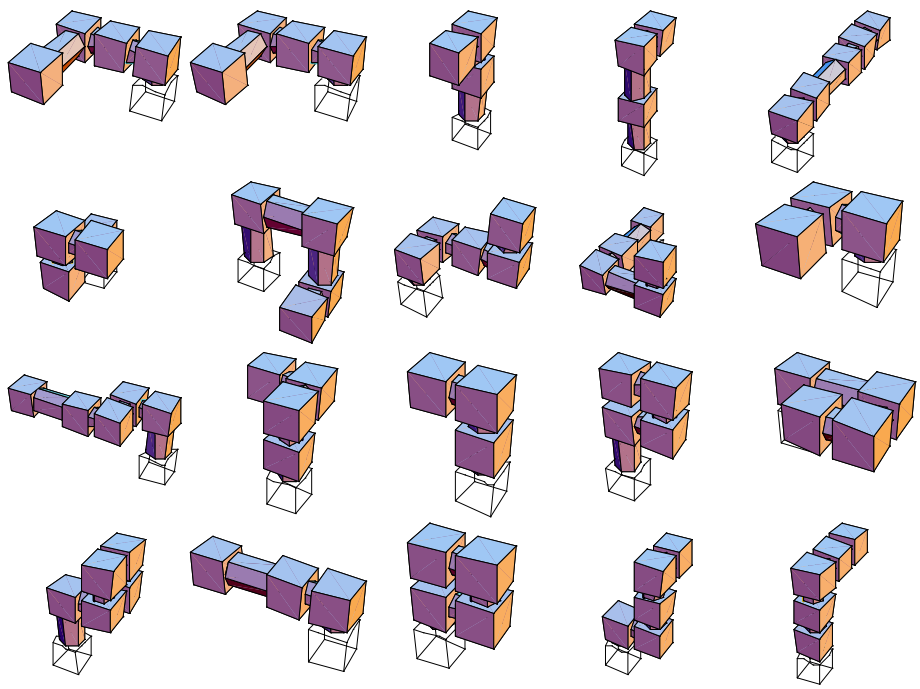


Figure 7.7: Robot configurations in the initial generation (position problem)

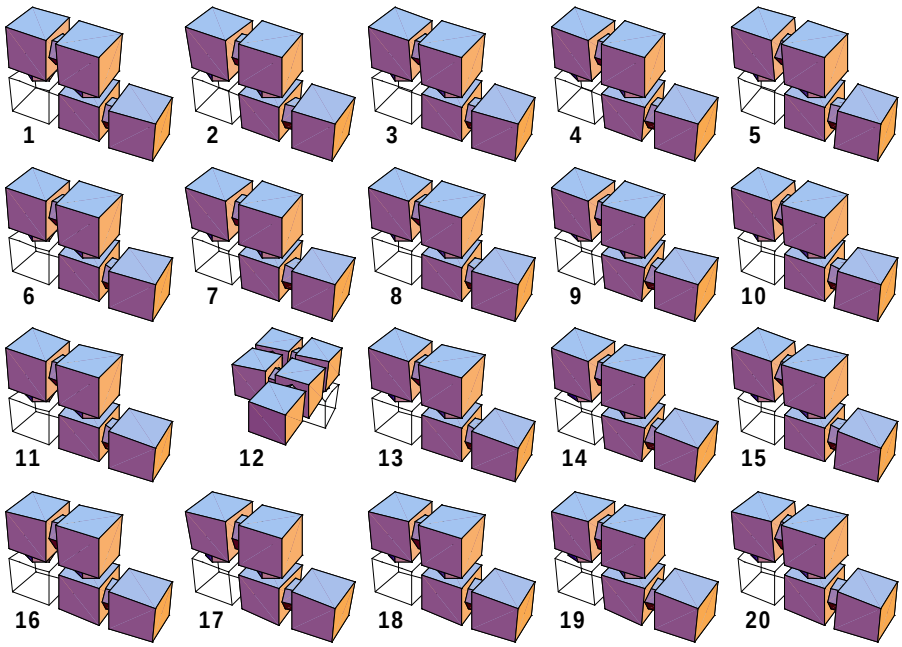


Figure 7.8: Robot configurations in the final generation (position problem)

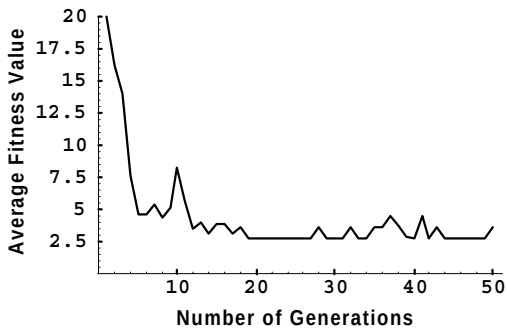


Figure 7.9: Average fitness value in every generation (position problem)

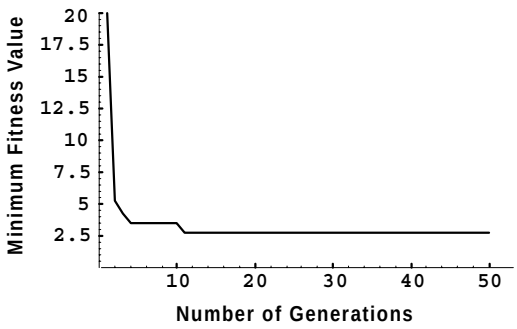


Figure 7.10: Minimal fitness value in every generation (position problem)

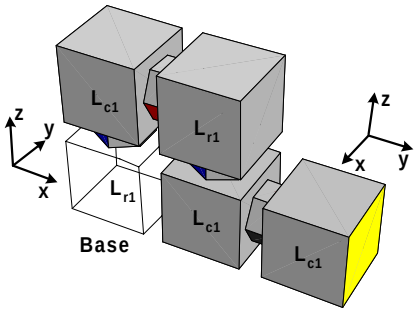


Figure 7.11: Optimal robot configuration (position problem)

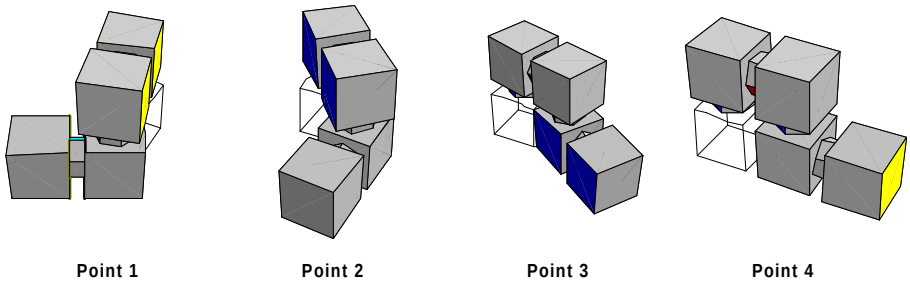


Figure 7.12: Execution of the given task with the optimal robot configuration (position problem)

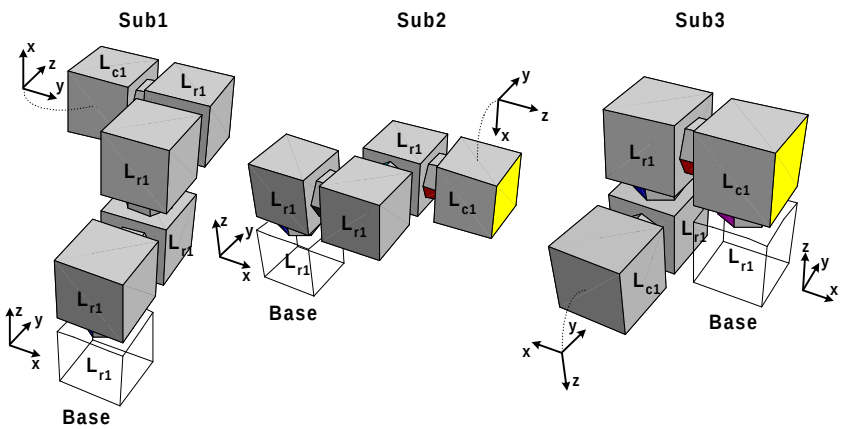


Figure 7.13: Sub-optimal robot configurations (position problem)

The computation results are shown in Fig. 7.7, 7.8, 7.9, and 7.10. Fig. 7.7 shows the initial generation of the robot configurations which consists of 20 different robot configurations. As shown in Fig. 7.9 and 7.10, both the average and minimal fitness values in the initial generation are equal to 20.0 which indicates that none of the initial generation of robot configurations is feasible. From the second generation, both average and minimal fitness values begin to decrease indicating that at least one feasible configuration has been found. When the destination generation, the 50th generation, is arrived, the average and minimum fitness values become 3.61 and 2.75 respectively. The final generation of robot configurations is shown in Fig. 7.8. All these configurations, except for the 12nd configuration (which is an infeasible one), have the same kinematic structure. Each of them contains two revolute joint modules (2 DOFs), three cubic link modules, and one virtual module, and therefore has the fitness of 2.75 (the smallest fitness in all the 50 generations). Note that due to the simple schematic drawing routine, the cubic link module L_{c1} and revolute link module L_{r1} look alike. However, we can identify them through the resulting AIMs. The AIM of the optimal robot configuration is given by

$$\mathcal{A}_{\mathbf{opt}} = \begin{bmatrix} (z,-x) & 0 & 0 & 0 & 0 & L_{r1} \\ (-x,-z) & (z,-y) & 0 & 0 & 0 & L_{c1} \\ 0 & (-x,y) & (z,x) & 0 & 0 & L_v \\ 0 & 0 & (-x,-y) & (z,-x) & 0 & L_{r1} \\ 0 & 0 & 0 & (-x,-z) & (z,-y) & L_{c1} \\ 0 & 0 & 0 & 0 & (-y,x) & L_{c1} \\ J_{r1} & J_{f1} & J_v & J_{r1} & J_{f1} & 0 \end{bmatrix}.$$

The execution of the given task (four points) with the optimal robot configuration is shown in Fig. 7.12. Three sub-optimal assembly configurations are also obtained as shown in Fig. 7.13. They have the fitness values of 5.25 (5 DOFs), 4.25 (4 DOFs), and 3.5 (3 DOFs) respectively. Their corresponding AIMs are given by

$$\mathcal{A}_{\mathbf{sub1}} = \begin{bmatrix} (z,-y) & 0 & 0 & 0 & 0 & L_{r1} \\ (x,-z) & (z,y) & 0 & 0 & 0 & L_{r1} \\ 0 & (-x,y) & (z,-x) & 0 & 0 & L_{r1} \\ 0 & 0 & (x,-z) & (z,y) & 0 & L_{r1} \\ 0 & 0 & 0 & (y,z) & (z,x) & L_{r1} \\ 0 & 0 & 0 & 0 & (y,-x) & L_{c1} \\ J_{r1} & J_{r1} & J_{r1} & J_{r1} & J_{r1} & 0 \end{bmatrix};$$

$$\mathcal{A}_{\mathbf{sub2}} = \begin{bmatrix} (z,-y) & 0 & 0 & 0 & 0 & L_{r1} \\ (-x,y) & (z,-x) & 0 & 0 & 0 & L_{r1} \\ 0 & (-x,y) & (z,-x) & 0 & 0 & L_v \\ 0 & 0 & (x,-y) & (z,x) & 0 & L_{r1} \\ 0 & 0 & 0 & (y,-z) & (z,x) & L_{r1} \\ 0 & 0 & 0 & 0 & (-z,-x) & L_{c1} \\ J_{r1} & J_{r1} & J_v & J_{r1} & J_{r1} & 0 \end{bmatrix};$$

$$\mathcal{A}_{\mathbf{sub3}} = \begin{bmatrix} (z,-y) & 0 & 0 & 0 & 0 & L_{r1} \\ (-x,-y) & (z,x) & 0 & 0 & 0 & L_{c1} \\ 0 & (-x,y) & (z,x) & 0 & 0 & L_v \\ 0 & 0 & (-x,-z) & (z,-y) & 0 & L_{r1} \\ 0 & 0 & 0 & (y,z) & (z,x) & L_{r1} \\ 0 & 0 & 0 & 0 & (y,-x) & L_{c1} \\ J_{r1} & J_{f1} & J_v & J_{r1} & J_{r1} & 0 \end{bmatrix}.$$

Example 7.2: Pose Design Problem

Now we wish to design a MDOF robot that can pass a set of task poses listed in Table 7.3. In this example, we set $n_m = 6$ and $n_g = 100$. The mutation operators for the search algorithm are listed in Table 7.4.

Table 7.3: Task poses

Pose 1	Pose 2
$\begin{bmatrix} -0.433 & 0.866 & 0.250 & -478.11 \\ 0.750 & -0.500 & -0.433 & -128.11 \\ -0.500 & 0 & -0.866 & -256.22 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.500 & 0.707 & -0.500 & -597.49 \\ 0.500 & -0.707 & -0.500 & -102.52 \\ -0.707 & 0 & -0.707 & -144.98 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Pose Three	**Pose Four**
$\begin{bmatrix} 0.268 & 0.500 & -0.824 & -751.57 \\ 0.155 & -0.866 & -0.475 & -29.76 \\ -0.951 & 0 & -0.309 & 133.68 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & -1 & -700 \\ 0 & -1 & 0 & 350 \\ -1 & 0 & 0 & 350 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

The computation results are shown in Fig. 7.14, 7.15, 7.16, and 7.17. Fig. 7.14 shows the initial generation of robot configurations which contains 20 different robot configurations, and none of them is feasible because both minimal and average fitness values are equal to 20 (Fig. 7.16 and 7.17). From the 22nd generation as shown in Fig. 7.16 and 7.17, both the average and minimal fitness values begin to decrease, which indicates that at least one feasible configuration has been found. When the destination generation, the

Table 7.4: The adaptive probabilities for the mutation operators (pose problem)

Average Fitness Value F_a	Link mutation Operator	Port Mutation Operator
20	0.2	0.2
$10 \leq F_a < 20$	0.06	0.045
$5 \leq F_a < 10$	0.045	0.03
$3 \leq F_a < 5$	0.035	0.02
$F_a < 3$	0.03	0.015

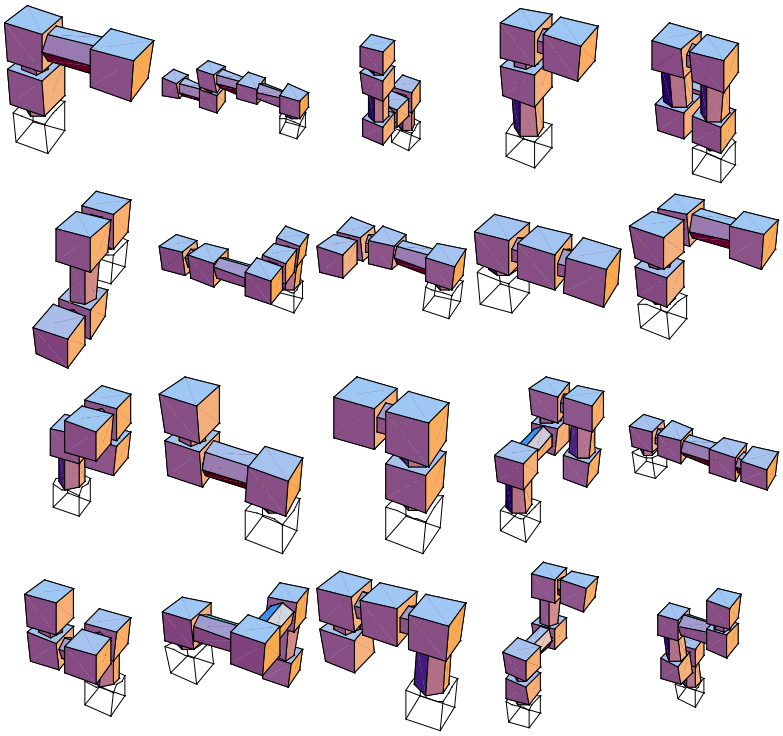


Figure 7.14: Robot configurations in the initial generation (pose problem)

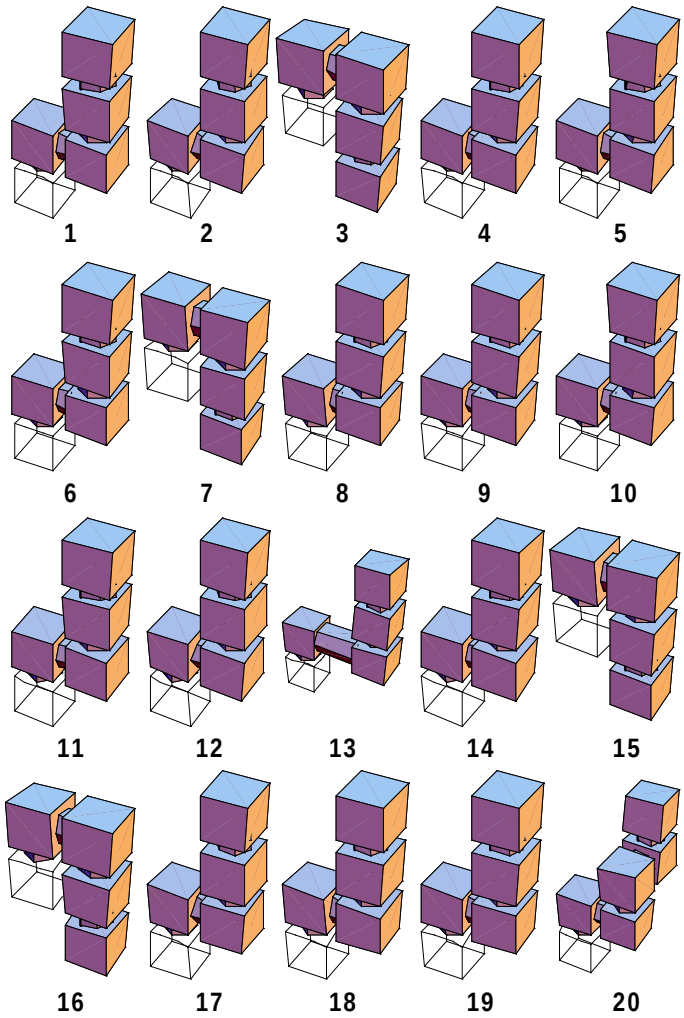


Figure 7.15: Robot configurations in the final generation (pose problem)

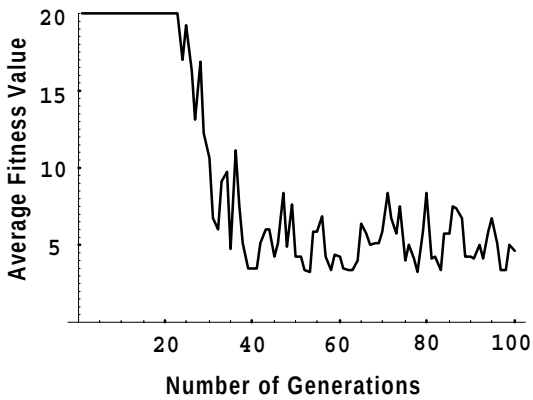


Figure 7.16: Average fitness value in every generation (pose problem)

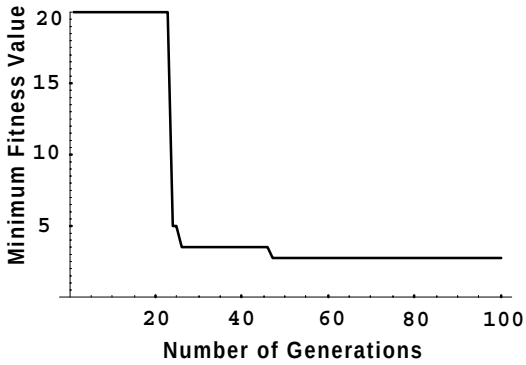


Figure 7.17: Minimal fitness value in every generation (pose problem)

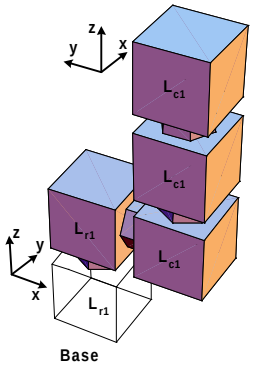


Figure 7.18: Optimal robot configuration (pose problem)

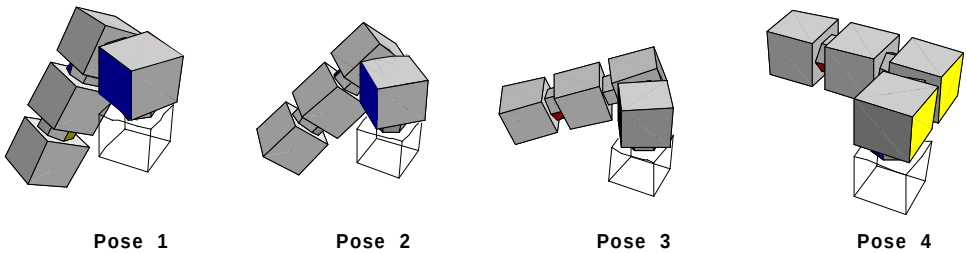


Figure 7.19: Execution of the given task with the optimal robot configuration (pose problem)

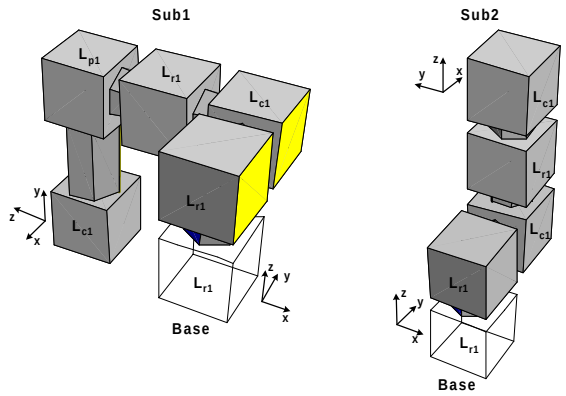


Figure 7.20: Sub-optimal robot configurations (pose problem)

100th generation, is reached, the average and minimal fitness values become 4.63 and 2.75 respectively. The robot configurations in the final generation are shown in Fig. 7.15. Both the 13rd and 20th robot configurations in Fig. 7.15 have the fitness value of 20 and are thus infeasible ones. The 3rd,7th, 15th, and 16th robot configurations have the same kinematic structure and bear the fitness value of 3.5. From the AIM, we can see that this configuration contains three revolute link modules (3 DOFs), two cubic link modules, and one virtual link module. The reminders belong to another set of kinematically identical configurations with each of them consisting of two revolute link modules (2 DOFs), three cubic link modules, and one virtual link module, and having the lowest fitness value of 2.75, and hence become the optimal one (Fig. 7.18). The optimal AIM is

$$\mathcal{A}_{\mathbf{opt}} = \begin{bmatrix} (z,x) & 0 & 0 & 0 & 0 & L_{r1} \\ (-x,z) & (z,y) & 0 & 0 & 0 & L_{r1} \\ 0 & (x,y) & (z,-y) & 0 & 0 & L_v \\ 0 & 0 & (-z,-y) & (z,x) & 0 & L_{c1} \\ 0 & 0 & 0 & (-z,-y) & (z,x) & L_{c1} \\ 0 & 0 & 0 & 0 & (-z,x) & L_{c1} \\ J_{r1} & J_{r1} & J_v & J_{f1} & J_{f1} & 0 \end{bmatrix}.$$

The execution of the given task (four poses) with the optimal robot configuration is shown in Fig. 7.19. Two sub-optimal assembly configurations are also obtained from the intermediate generations as shown in Fig. 7.20, which have the fitness values of 5.0 (4 DOFs) and 3.5 (3 DOFs) respectively. Their corresponding AIMs are given by

$$\mathcal{A}_{\mathbf{sub1}} = \begin{bmatrix} (z,y) & 0 & 0 & 0 & 0 & L_{r1} \\ (y,z) & (z,-y) & 0 & 0 & 0 & L_{r1} \\ 0 & (-x,-y) & (z,-y) & 0 & 0 & L_{c1} \\ 0 & 0 & (-z,-y) & (z,y) & 0 & L_{r1} \\ 0 & 0 & 0 & (y,z) & (z,-x) & L_{p1} \\ 0 & 0 & 0 & 0 & (y,x) & L_{c1} \\ J_{r1} & J_{r1} & J_{f1} & J_{r1} & J_{p1} & 0 \end{bmatrix};$$

$$\mathcal{A}_{\mathbf{sub2}} = \begin{bmatrix} (z,y) & 0 & 0 & 0 & 0 & L_{r1} \\ (-y,z) & (z,-y) & 0 & 0 & 0 & L_{r1} \\ 0 & (x,y) & (z,-y) & 0 & 0 & L_{v} \\ 0 & 0 & (-x,-z) & (z,x) & 0 & L_{c1} \\ 0 & 0 & 0 & (-z,-y) & (z,y) & L_{r1} \\ 0 & 0 & 0 & 0 & (-z,-x) & L_{c1} \\ J_{r1} & J_{r1} & J_{v1} & J_{f1} & J_{r1} & 0 \end{bmatrix}.$$

Remark 1: In this section, we present two computation examples for demonstration purpose. The computation results are quite encouraging because the optimal robot configurations in both "position" and "pose" design examples are kinematically identical to the configurations used to generate the given task points and poses respectively. We can also geometrically show that each of the derived optimal MDOF robots (2 DOFs) contains the minimal DOFs. Taking the "position" design example for instance, four given task points (Table 7.1) do not fall in a circle or a straight line. Hence, it would require at least two DOFs to perform this task.

Remark 2: The computation results also show that the reachability constraint has significant effect on the search efficiency. In the position design example, the first feasible configuration is derived in the second generation and the optimal configuration first appears in the 11st generation as shown in Fig. 7.10. In the pose design example, however, the first feasible configuration is derived in the 22nd generation and the optimal configuration first appears in the 47th generation as shown in Fig. 7.17. This is due to fact that the positional reachability constraint is easier to be satisfied. The pose design example takes about 6 hours computation time on a SUN ULTRASPARC 1 workstation, while the position design example takes only 2 hours computation time under the same environment.

Remark 3: It is the consideration of virtual modules that allows the algorithm to perform both type and dimension synthesis uniformly. To a certain extent, the search for optimal

robot configuration is a procedure to increase the number of virtual and cubic link modules in the candidate configurations.

Remark 4: In the evolutionary algorithm, there are several user-defined control parameters, such as the population size, mutation and crossover operators, which have significant effect on the search efficiency. Many trial computations have been done in order to draw the suitable values to be used for the control parameters. (1) The suitable population size is between 15 to 40; (2)the link and port mutation operators are between 0.01 to 0.06; (3) the crossover operator has little effect on the search efficiency and the recommended range is between 0.2 to 0.4. The algorithm, as demonstrated by the provided examples, is reliable and effective under the suitable control parameters.

7.6 Discussion

A modular reconfigurable robot system has the advantage of providing an optimal robot configuration for a specific task. From the application point of view, a MDOF modular robot configuration is optimal only if it can perform the given task. The design of MDOF modular robot configurations is formulated as a discrete design optimization problem. The design parameters are defined on the AIMs. The objective function is concentrated on reducing the DOFs of a robot. Several performance constraints such as the task related constraints and the mechanical constructability constraint are considered to ensure the feasibility of a MDOF modular robot configuration. An evolutionary algorithm, based on the effective coding schemes, is employed to search the optimal solution. A virtual module concept is also introduced in the coding schemes to guarantee the operation of EA. The algorithm, as demonstrated by the provided examples, is reliable and effective under suitable control parameters. Since the EA approach follows probabilistic rules, the optimal solution is not guaranteed within a fixed number of generations, but sub-optimal solutions can be always attained.

Chapter 8

Conclusions

In this thesis, we have studied the fundamental research issues of a modular reconfigurable robot system (MRRS) including the conceptual module design, modular robot assembly representation, and algorithms for kinematics, dynamics, calibration, and robot configuration optimization. In this chapter, we first summarize the main contributions, and then outline the future research directions.

8.1 Contributions

To develop a modular reconfigurable robot system, we made some contributions in the following aspects:

- **Conceptual Module Design**

  The individual types of modules considered include 1-DOF revolute joint modules, 1-DOF prismatic joint modules, and cubic link modules. For a diversity of task requirements, each type of module comes with different sizes. A set of connectors has also been designed for both mechanical and electrical connections. Each module is designed as a cube with multiple connecting sockets so that many possible configurations can be constructed with an inventory of modules. This design philosophy provides a direction for us to design a flexible MRRS using minimal types of modules.

- **Modular Robot Assembly Representation**

Based on the graph theory and the AIM representation of modular robot assembly configurations [14, 17], the accessibility matrix and path matrix concepts are adopted for indicating the topological structures of tree-structured modular robots. Algorithms that can automatically derive the accessibility matrix and path matrix from a given AIM have also been developed. The matrix representation methods pave the way for the development of configuration-independent algorithms.

- **Modular Robot Kinematics**

  Based on the local frame representation of POE formula, the forward and inverse kinematic algorithms are proposed for tree-structured modular robots. Both algorithms are configuration independent such that once an AIM is given, the forward and inverse kinematic models can be generated automatically. The kinematic transformation between two consecutive modules is formulated as dyad kinematics. Applying the dyad kinematics recursively from the base to each of the pendant links, the forward kinematics of a general tree-structured robot can be derived. The inverse kinematic algorithm, on the other hand, follows a numerical approach in which the body manipulator Jacobian is formulated in terms of linear operation on $se(3)$. The differential change of the end link pose is defined as the matrix logarithm on $SE(3)$. With some modifications, the algorithm can deal with not only the general inverse problem in which the desired end link poses are given by elements of $SE(3)$, but also the pure position (3×1 vectors), pure orientation (elements of $SO(3)$), and the hybrid inverse problem. Computation results show that this algorithm is robust and efficient. However, this algorithm, like other numerical methods, cannot guarantee that a solution can be derived within a fixed number of iterations. The computation efficiency depends on the initial guess solution.

- **Modular Robot Dynamics**

  Both inverse and forward dynamic problems are studied. Like the kinematic modeling techniques, the proposed inverse and forward dynamic algorithms are also configuration-independent and suitable for tree-structured modular robots. Given an AIM, the dynamic equations of motion can be formulated in both the recursive form and the closed form automatically. The formulation of the dynamic model

starts with the modified recursive Newton-Euler algorithm in which the generalized velocities, accelerations, and forces (wrenches) are expressed in terms of $se(3)$. The recursive dynamic model is then constructed into a closed form by means of the accessibility matrix. The recursive dynamic model is computationally efficient ($O(n)$). However, many applications still depend on explicit closed-form expression of the equations of motion such as robot design, system identification, motion optimization, and optimal control. A numerical forward dynamic algorithm, based on the Runge-Kutta method, is employed for the system simulation purpose.

- **Kinematic Calibration for Modular Robots**

  A novel kinematic calibration algorithm is proposed. This algorithm is based on the local frame representation of POE formula and differential geometry. The kinematic errors are attributed to the kinematic errors in the initial poses in the dyads, which significantly simplify the calibration model and provide a clear geometric interpretation of the calibration process. A least-square algorithm is employed to iteratively solve the calibration result. Simulation examples have shown that the calibration algorithm is simple and robust for modular robot applications. This algorithm is also configuration independent, and can be applied to industrial robots without modification.

- **Optimization of Modular Robot Configurations**

  The Minimized Degree-of-freedom (MDOF) concept is introduced in optimizing modular robot configurations for a specific task requirement. With fewer numbers of modules and DOFs, a modular robot would have a simple configuration and a low power consumption rate, so that it can perform the task effectively. Due to the discrete module set, the design of MDOF modular robot configurations is formulated as a combinatorial optimization problem. The design parameters are defined on the AIMs. The objective function is concentrated on minimizing the number of DOFs of a robot. Several performance constraints such as the task related constraints and mechanical constructability constraints are considered to ensure the feasibility of MDOF modular robot configurations. An evolutionary algorithm (EA), based on effective coding schemes, is employed to search for the optimal solution. A virtual

module concept is also introduced in the coding schemes to guarantee the operation of EA. The algorithm, as demonstrated by examples, is reliable and effective under suitable control parameters. Since the EA approach follows probabilistic rules, the optimal solution is not guaranteed within a fixed number of generations, but sub-optimal solutions can be always attained.

8.2 Future Directions

Although the basic issues pertaining to the MRRS have been studied, there are still a number of unexplored topics in this area.

- **Improvement on the Module Design**

  In our present module design, the cubic link modules and connectors/adapter are designed individually. However, if a longer link is required for a specific application, a number of cubic link modules are needed to form the link with required dimensions which may make the link heavy and lower the loading capability of the robot. A proposed design methodology is to combine the designs of cubic link modules and connectors together to form a set of new connectors with various sizes and geometries. The newly designed connectors will act as both links and connectors. Since each of the joint modules has a cubic or cube-like shape, the new connectors are not necessary to be designed as cubes. By doing so, the modular robot system will be more practical and cost-effective. In order to rapidly assemble the robot modules, another modification on the present design is to employ or design a simple and reliable quick-coupling mechanism for the module connections.

- **Closed-form Inverse kinematics Solution for MDOF Robots**

  Although the proposed numerical inverse kinematics algorithm is suitable for various modular robots regardless of the joint types and the number of DOFs, it cannot guarantee a feasible solution to be derived within a fixed number of iterations. As it is employed for on-line computation, this algorithm is not very reliable except for some special applications in which the initial guess solutions are very close to

the real solutions, for example, the seam tracking application. Hence, to derive closed-form solutions for modular robots systematically is very important in order to increase the computational efficiency. Unfortunately, whether the inverse solution can be expressed in a closed form or not would depend on the configuration of the robot, e.g., the number of DOFs, the joint types, and their corresponding order (e.g., RRRRRR, PPPRRR, RRRP, ...), and no such formulation is known for the general case.

As mentioned previously, our goal is to design the MDOF robot configurations for given tasks. Most of the MDOF robots have simple configurations with serial topology and minimal DOFs. Due to our module design, all the adjacent joint axes of the modular robots are either parallel or perpendicular to each other. This type of arrangement usually results in simplified kinematic solutions. It is, therefore, possible to derive the closed-form solution for a modular robot with DOFs less than six, based on the POE formula. This method might not be generic, but should be applicable for the inverse solutions of most modular robots.

- **Hybrid Kinematics and End-effector Design**

  The term *hybrid kinematics* is referred to as the combination of robot manipulator kinematics and the end-effector kinematics. In order to further minimize the number of modules as well as DOFs used in a modular robot, synthesizing the motion of the end-effector in conjunction with the motion of the robot arm is necessary. The number of the actuating units in a complete robotic workcell can be minimized by combining the motions generated by the robot joint modules and the end-effector. The hybrid kinematics could also serve as a guideline for the design of the end-effector.

- **Comprehensive Algorithm for the Design of MDOF Modular Robots**

  To a certain extent, the proposed algorithm on optimal design of modular robot configurations is only a preliminary work. Future research is required to investigate this problem.

1. *Hierarchical Task Planner:* The current definition of robot task admits a low level task description, i.e., a set of discrete task points or poses in the workspace. For the user, however, it would be more convenient to provide a high level task description, e.g., "assemble this PCB". In order to determine an optimal robot configuration to handle such a high level task description, one would need a hierarchical task planner which can translate the high-level task commands into low level descriptions. Such a planner as mentioned by Paredis [93] will use the kinematic structure of the robot as a constraint so that there exists a strong coupling between the task planner and the optimization algorithms. Therefore, it may be a feasible approach to integrate both the planner and the optimization algorithms into a single framework.

2. *Dynamic Design Criteria:* Our optimization algorithm is focused on the kinematic performance of robots. A comprehensive algorithm is also required to consider the dynamic performance of robots. A practical approach is to consider the dynamic performance measures as constraints, such as joint velocity, acceleration, and torque limitations. However, since these constraints will rely on the trajectory planning algorithm, a suitable trajectory planning algorithm is to be embodied into the optimization algorithm.

Our ultimate goal is to provide "local intelligence" to every module and each module will be designed as a self-contained, intelligent unit with its own controller and with sufficient computation and communication capability. It can work individually or collectively. If it works individually, it can be used as an individual intelligent motion control unit. If these modules are connected together, they can collectively determine the entire robot assembly configuration and the control scheme for this configuration. They can also determine how to reconfigure themselves into one or several optimal robot configurations to perform a given task in coordination. This goal will bring a modular reconfigurable robot system closer to a distributed, intelligent, and rapidly deployable metamorphic automation system. Such a system will inherit merits from both robotics and hard automation such as high precision, high agility, high operational efficiency, large working envelope, wide application spectrum, and open control architecture. Potential application

areas of this metamorphic automation system could be in structured or unstructured and unpredictable environments, entertainment industries, and service industries. Numerous applications can be explored through the design of specialized modular units and the reconfiguration of the available modules.

Appendix A

Generalized Mass Matrix for a Link Assembly

In the dynamic formulation algorithm (Chapter 5), we consider each link (except for the base link) and its input joint (connector/adapter) as a rigid body, termed link assembly. An $(n+1)$-module tree-structured robot thus consists of n link assemblies. According to the module specification, we are able to calculate the generalized mass matrix for each link assembly when the link (module) type, the input joint type, and the joint approaching direction (connecting port vector) are known. This appendix proposes a computation scheme that can derive the generalized mass matrix for every link assembly from the given AIM automatically.

A.1 Coordinate System in a Link Assembly

As shown in Fig. A.1, we assign five body-fixed frames in each link assembly, i.e., frame i - the link module frame, frame i^m - the CM frame of link module, frame j - the connector frame, frame j_m - the CM frame of the connector, and frame ij^m - the CM frame of the whole link assembly. For convenience, we let frame i, i^m, and ij^m be parallel to each other and frame j^m be parallel with j. Based on the symmetric design of modules and connectors, the origin of frame i^m and j^m will be located at the z-axis of frame i and j respectively.

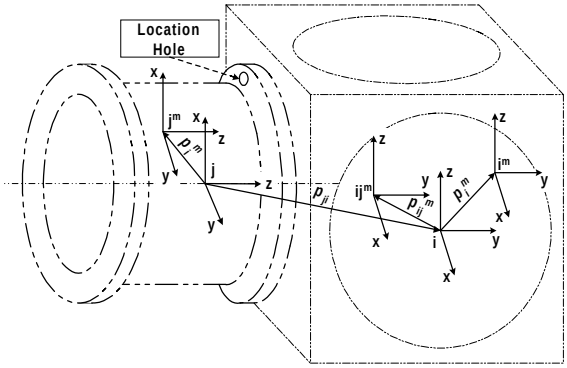


Figure A.1: Coordinate system in a link assembly

A.2 Kinematic Transformation from Frame j to i

To compute the generalized mass matrix of a link assembly, the kinematic transformation from the connector frame j to link module frame i is to be determined. Since the x or y axis of the connector frame can be arbitrarily assigned, we set the x-axis to point out to the direction of the location hole (as shown in Fig. 2.9). Like a link module, we also define a connecting port vector on the connector, and thus the connecting port vector will be (z, x). Therefore, the kinematic transformation from frame j to i is the same as the dyad kinematics mentioned in Chapter 3, and can also be derived from the given AIM automatically. Here, we denote the kinematic transformation from frame j to i by $T_{j,i} = (p_{ji}, R_{ji})$. The kinematic transformation from frame j^m to i can be given by: $T_{j^m,i} = T_{j^m,j}T_{j,i} = (p_{ji} - p_{j^m}, R_{ji})$, where p_{j^m} is the position vector of frame j^m with respect to frame j. It follows that $T^{-1}_{j^m,i} = (R^T_{j,i}(p_{j^m} - p_{ji}), R^T_{ji})$=$T_{i,j^m}$=$(p_{i,j^m}, R_{i,j^m})$.

A.3 Determination of the CM of a Link Assembly

The CM frame of a link assembly ij^m is supposed to be parallel with the link module frame i. The origin of this frame with respect to module frame i is given by:

$$p_{ij^m} = \frac{m_j p_{ij^m} + m_i p_{i^m}}{m_i + m_j} \tag{A.1}$$

$$= \frac{m_j R^T_{ji}(p_{j^m} - p_{ji}) + m_i p_{i^m}}{m_i + m_j} \tag{A.2}$$

where m_i and m_j are the mass of the link module and connector respectively; p_{i^m} is the position vector of frame i^m with respect to frame i.

The kinematic transformation from frame i^m to frame ij^m thus can be written as

$$\begin{aligned} T_{i^m,ij^m} &= T_{i^m,i}T_{i,ij^m} \\ &= T_{i,i^m}^{-1}T_{i,ij^m} \\ &= (p_{ij^m} - p_{i^m}, I) \\ &= (p_i, R_i) \end{aligned} \tag{A.3}$$

where $p_i = p_{ij^m} - p_{i^m}$ and $R_i = I$.

The kinematic transformation from frame j^m to frame ij^m thus can be written as

$$\begin{aligned} T_{j^m,ij^m} &= T_{j^m,i}T_{i,ij^m} \\ &= (R_{ji}p_{ij^m} + p_{ji} - p_{j^m}, R_{ji}) \\ &= (R_{ji}p_{ij^m} + p_{ji} - p_{j^m}, R_{ji}) \\ &= (p_j, R_j) \end{aligned} \tag{A.4}$$

where $p_j = R_{ji}p_{ij^m} + p_{ji} - p_{j^m}$ and $R_j = R_{ji}$.

A.4 Generalized Mass Matrix of a Link Assembly

Let M_{i^m} represent the generalized mass matrix of a link module expressed in its own CM frame, i^m, then it has the form of

$$M_{i^m} = \begin{bmatrix} m_i I & 0 \\ 0 & J_i \end{bmatrix}. \tag{A.5}$$

where I is a 3×3 identity matrix; J_i is the inertia matrix of the link module with respect to its own CM frame (i^m), which can be found from Appendix A.

According to Equation 5.12, the generalized mass matrix of the link module with respect to the CM frame of the link assembly, ij^m, can be given by

$$\begin{aligned} M_i &= Ad^T_{T_{i^m,ij^m}} M_{i^m} Ad_{T_{i^m,ij^m}} \\ &= \begin{bmatrix} m_i I & m_i \hat{p}_i \\ -m_i \hat{p}_i & J_i - m_i \hat{p}_i^2 \end{bmatrix} \end{aligned} \tag{A.6}$$

where $\hat{p}_i$ is the skew-symmetric matrix related to p_i.

Let M_{j^m} represent the generalized mass matrix of a joint (connector/adapter) expressed in its own CM frame, j^m, then it also has the form of

$$M_{j^m} = \begin{bmatrix} m_j I & 0 \\ 0 & J_j \end{bmatrix}. \tag{A.7}$$

where J_j is the inertia matrix of the connector/adapter with respect to its own CM frame (j^m), which can also be found from Appendix A.

The generalized mass matrix of the connector with respect to the CM frame of the link assembly, ij^m, can be given by

$$M_j = Ad^T_{T_{j^m,ij^m}} M_{j^m} Ad_{T_{j^m,ij^m}} \tag{A.8}$$

$$= \begin{bmatrix} m_j I & m_j R_j^T \hat{p}_j R_j \\ -m_j R_j^T \hat{p}_j R_j & R_j^T (J_j - m_j \hat{p}_j^2) R_j \end{bmatrix} \tag{A.9}$$

where $\hat{p}_j$ is the skew-symmetric matrix related to p_j.

Therefore, the generalized mass matrix of the link assembly with respect to the CM frame of the link assembly can be given by

$$M_{ij} = M_i + M_j \tag{A.10}$$

Appendix B

Geometric Interpretation of the Kinematic Errors

As shown in Fig. B.1, let $T^c_{i-1,i}(0)$ be the initial pose of the calibrated module frame i^c with respect to module frame $(i-1)$. Due to the geometric errors in the dyad, the nominal module frame i is different from the calibrated module frame i^c. The differential change of $T_{i-1,i}(0)$, denoted by $dT_{i-1,i}(0)$, is given by

$$dT_{i-1,i}(0) = T^c_{i-1,i}(0) - T_{i-1,i}(0). \tag{B.1}$$

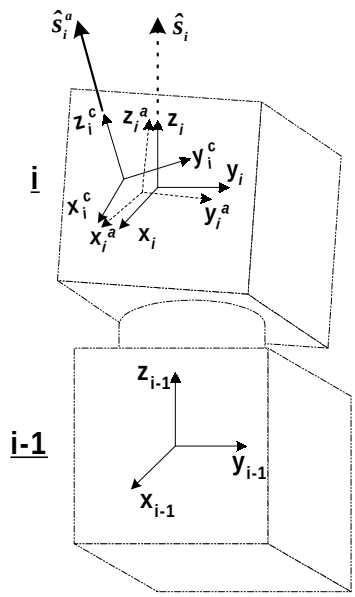


Figure B.1: Kinematic errors in a dyad

Based the differential transformation theory, frame i ($T_{i-1,i}(0)$) can be kinematically transformed into frame i^c ($T^c_{i-1,i}(0)$) under an infinitesimal translation **Trans** (dx_i, dy_i, dz_i)

followed by an infinitesimal rotation, **Rot** (δx_i, δy_i, δz_i), where dx_i, dy_i, and dz_i are infinitesimal displacements along x , y , and z-axis of frame i respectively, and δx_i, δy_i, and δz_i are infinitesimal angles about x, y, and z-axis of frame i respectively. Then

$$T^c_{i-1,i}(0) = T_{i-1,i}(0)\mathbf{Trans}(dx_i, dy_i, dz_i)\mathbf{Rot}(\delta x_i, \delta y_i, \delta z_i). \tag{B.2}$$

Note that the differential transformation is expressed in frame i. Hence, Equation (B.2) follows the right multiplicative differential transformation of $T_{i-1,i}(0)$ [108, 89].

Based on the definition of the differential transformation, we have

$$\mathbf{Trans}(dx_i, dy_i, dz_i) = \begin{bmatrix} 1 & 0 & 0 & dx_i \\ 0 & 1 & 0 & dy_i \\ 0 & 0 & 1 & dz_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{B.3}$$

$$\mathbf{Rot}(\delta x_i, \delta y_i, \delta z_i) = \begin{bmatrix} 1 & -\delta z_i & \delta y_i & 0 \\ \delta z_i & 1 & -\delta x_i & 0 \\ -\delta y_i & \delta x_i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{B.4}$$

Substituting Equation (B.3),(B.4), and (B.2) into Equation (B.1), we get

$$dT_{i-1,i}(0) = T_{i-1,i}(0)\left[\mathbf{Trans}(dx_i, dy_i, dz_i)\mathbf{Rot}(\delta x_i, \delta y_i, \delta z_i) - I\right] \tag{B.5}$$

$$= T_{i-1,i}(0)d\hat{p}_i, \tag{B.6}$$

where I represents the 4×4 identity matrix; and

$$\begin{aligned} d\hat{p}_i &= [\mathbf{Trans}(dx_i, dy_i, dz_i)\mathbf{Rot}(\delta x_i, \delta y_i, \delta z_i) - I] \\ &= \begin{bmatrix} 0 & -\delta z_i & \delta y_i & dx_i \\ \delta z_i & 0 & -\delta x_i & dy_i \\ -\delta y_i & \delta x_i & 0 & dz_i \\ 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{B.7}$$

$d\hat{p}_i$ represents the kinematic error in the initial pose of module frame i ($T_{i-1,i}(0)$), observed in frame i. It is apparent that $d\hat{p}_i$ is an element of $se(3)$ and can be identified by a 6×1 vector so that $d\hat{p}_i \mapsto dp_i = (dx_i,dy_i,dz_i,\delta x_i,\delta y_i,\delta z_i)^T$. Equation (B.6) also implies that

$$d\hat{p}_i = T^{-1}_{i-1,i}(0)dT_{i-1,i}(0). \tag{B.8}$$

Equations (B.6) and (B.8) are formulated by performing the differential transformation with respect to module frame i. Similarly, we can also perform the differential transformation with respect to module frame $i-1$. It follows that the module frame i ($T_{i-1,i}(0)$)

can also be kinematically transformed into the calibrated frame i^c ($T_{i-1,i}(0)^c$) under an infinitesimal rotation $\mathbf{Rot}(\delta x_{i-1,i},\ \delta y_{i-1,i},\ \delta z_{i-1,i})$ and followed by an infinitesimal translation $\mathbf{Trans}(dx_{i-1,i},\ dy_{i-1,i},\ dz_{i-1,i})$, where $\delta x_{i-1,i}$, $\delta y_{i-1,i}$, and $\delta z_{i-1,i}$ are infinitesimal rotation angles about x, y, and z-axis of frame $i-1$ respectively, and $dx_{i-1,i}$, $dy_{i-1,i}$, and $dz_{i-1,i}$ are infinitesimal displacements along x , y , and z-axis of frame $i-1$ respectively. It follows that

$$T_{i-1,i}(0)^c = \mathbf{Trans}(dx_{i-1,i}, dy_{i-1,i}, dz_{i-1,i})\mathbf{Rot}(\delta x_{i-1,i}, \delta y_{i-1,i}, \delta z_{i-1,i})T_{i-1,i}(0). \tag{B.9}$$

Note that the differential transformation is expressed in the module frame $i-1$, so that Equation (B.9) follows the left multiplicative differential transformation of $T_{i-1,i}(0)$ [108, 89].

Substituting Equation (B.9) into (B.1), we get

$$\begin{aligned} dT_{i-1,i}(0) &= [\mathbf{Trans}(dx_{i-1,i}, dy_{i-1,i}, dz_{i-1,i})\mathbf{Rot}(\delta x_{i-1,i}, \delta y_{i-1,i}, \delta z_{i-1,i}) - I]T_{i-1,i}(0) \\ &= d\hat{p}_{i-1,i}T_{i-1,i}(0), \end{aligned} \tag{B.10}$$

where

$$\begin{aligned} d\hat{p}_{i-1,i} &= [\mathbf{Trans}(dx_{i-1,i}, dy_{i-1,i}, dz_{i-1,i})\mathbf{Rot}(\delta x_{i-1,i}, \delta y_{i-1,i}, \delta z_{i-1,i}) - I] \\ &= \begin{bmatrix} 0 & -\delta z_{i-1,i} & \delta y_{i-1,i} & dx_{i-1,i} \\ \delta z_{i-1,i} & 0 & -\delta x_{i-1,i} & dy_{i-1,i} \\ -\delta y_{i-1,i} & \delta x_{i-1,i} & 0 & dz_{i-1,i} \\ 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{B.11}$$

$d\hat{p}_{i-1,i}$ represents the kinematic error in the initial pose of module frame i ($T_{i-1,i}(0)$), observed in module frame $i-1$. It is also an element of $se(3)$ and can be identified by a 6×1 vector so that $d\hat{p}_{i-1,i} \mapsto dp_{i-1,i} = (dx_{i-1,i},\ dy_{i-1,i},\ dz_{i-1,i},\ \delta x_{i-1,i},\ \delta y_{i-1,i},\ \delta z_{i-1,i})^T$. Equation (B.10) also implies that

$$d\hat{p}_{i-1,i} = dT_{i-1,i}(0)T^{-1}_{i-1,i}(0). \tag{B.12}$$

Consider the gross kinematic errors in a complete robot with n+1 modules and a tool. Similar to $T^{-1}_{i-1,i}(0)dT_{i-1,i}(0)$ and $dT_{i-1,i}(0)T^{-1}_{i-1,i}(0)$, the terms $T^{-1}_{0,n+1}dT_{0,n+1}$ and $dT_{0,n+1}T^{-1}_{0,n+1}$ represent the gross kinematic errors of a complete robot observed in the end-effector and the base frame respectively. They are also elements of $se(3)$.

Bibliography

[1] R.O. Ambrose. *Design, Construction and Demonstration of Modular, Reconfigurable Robots*. PhD thesis, University of Texas at Austin, U.S.A., 1991.

[2] R.O. Ambrose and D. Tesar. Modular robot connection design. In *Proceedings of ASME Conference on Flexible Assembly Systems*, pages 41–48, Scottsdate, AZ, 1992.

[3] J. Angeles. Numerical solution to the input output displacement equation of general $7R$ spatial mechanism. In *Proceedings of the Fifth World Congress and Mechanisms*, pages 1008–1011, 1979.

[4] J. Angeles. On the numerical solution of the inverse kinematics problem. *International Journal of Robotics Research*, 4(2):21–37, 1985.

[5] J. Angeles. The design of isotropic manipulator architectures in the presence of redundancies. *International Journal of Robotics Research*, 11(3):196–201, 1992.

[6] W. W. Armstrong. Recursive solution to the equation of motion in an n-link manipulator. In *Proceedings of the Fifth World Congress on Mechanisms and Machine Theory*, pages 1134–1346, 1979.

[7] B. Benhabib, R. Cohen, M.G. Lipton, and M.Q. Dai. Design of a rotary-joint-based modular robot. In *Proceedings of ASME Mechanism Conference*, pages 239–243, 1990.

[8] B. Benhabib, G. Zak, and M.G. Lipton. A generalized kinematic modeling method for modular robots. *Journal of Robotic Systems*, 6(5):545–571, 1989.

[9] R. Bernhardt and S.L. Albright. *Robot Calibration.* Chapman and Hall, London; New York, 1993.

[10] B. Bollobás. *Graph Theory - An Introductory Course.* Springer-Verlag, 1979.

[11] R. Brockett. Robotic manipulators and the product of exponential formula. In *International Symposium in Math. Theory of Network and Systems*, pages 120–129, Beer Sheba, Israel, 1983.

[12] R. Brockett, A. Stokes, and F.C. Park. A geometric formulation of the dynamic equation describing kinematic chains. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 637–641, Altanta, GA, May 1993.

[13] G. Chartrand and O. R. Oellermann. *Applied and Algorithmic Graph Theory.* McGraw-Hill, Inc., USA, 1993.

[14] I.M. Chen. *Theory and Applications of Modular Reconfigurable Robotic Systems.* PhD thesis, California Institute of Technology, CA, USA, 1994.

[15] I.M. Chen. On optimal configuration of modular reconfigurable robots. In *Proceedings of International Conference on Control, Automation, Robotics, and Vision*, Singapore, 1996.

[16] I.M. Chen and J.W. Burdick. Determining task optimal modular robot assembly configurations. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 132–137, Nagoya, Japan, May 1995.

[17] I.M. Chen and G. Yang. Configuration independent kinematics for modular robots. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1845–1849, 1996.

[18] I.M. Chen and G. Yang. Automatic generation of dynamics for modular robots with hybrid geometry. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 2288–2293, 1997.

[19] I.M. Chen and G. Yang. Automatic model generation for modular reconfigurable robot dynamics. *ASME Journal of Dynamic Systems, Measurement, and Control*, 120: 346-352, 1997.

[20] I.M. Chen and G. Yang. Geometric formulation of modular reconfigurable robot kinematics and dynamics. In *Proceedings of the 2nd Asian Control Conference*, pages 265–268, 1997.

[21] I.M. Chen and G. Yang. Kinematic calibration of modular reconfigurable robots using product-of-exponentials formula. *Journal of Robotic Systems*, 14(11):807–821, 1997.

[22] I.M. Chen and G. Yang. Inverse kinematics for modular reconfigurable robots. In *IEEE Conference on Robotics and Automation*, pages 1647–1652, 1998.

[23] I.M. Chen and G. Yang. An evolutionary algorithm for the reduction of DOFs in modular reconfigurable robots. In *Proceedings of the ASME Dynamic System and Control Division*, DSC-64: 759–766, 1998.

[24] I.M. Chen, G. Yang, and I.G. Kang. Numerical inverse kinematics for modular reconfigurable robots. *Journal of Robotic Systems*, 1999 (to appear).

[25] I.M. Chen, Yeo Song Huat, Chen Guang, and G. Yang. Kernel for modular robot applications - automatic modeling techniques. *International Journal of Robotics Research*, 18(2):225-242, 1999.

[26] I.M. Chen, G. Yang, C.T. Tan, and S.H. Yeo. A local POE model for robot kinematic calibration. Technical Report TP-MPE-NTU-01-98-01, School of MPE, Nanyang Technological University, Singapore, 1998.

[27] S.L. Chiu. Kinematic characterization of manipulators: an approach to defining optimality. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 828–833, Philadelphia, PA, April 1988.

[28] O. Chocron and P. Bidaud. Genetic design of 3d modular manipulator. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 223–228, 1997.

[29] R. Cohen, M.G. Lipton, M.Q. Dai, and B. Benhabib. Conceptual design of a modular robot. *ASME Journal of Mechanical Design*, 114:117–125, March 1992.

[30] J.J. Craig. *Introduction to Robotics Mechanics and Control.* Addison–Wesley, New York, USA, 2 edition, 1989.

[31] M.L. Curtis. *Matrix Groups.* Springer–Verlag, New York, USA, 2 edition, 1984.

[32] J. Denavit and R.S. Hartenberg. Kinematic notation for lower-pair mechanisms based on matrices. *ASME Journal of Applied Mechanics*, 2:215–211, 1955.

[33] J. Denavit and R.S. Hartenberg. An iterative method for the displacement analysis of special mechanisms. *ASME Journal of Applied Mechanics*, pages 309–314, 1964.

[34] N. Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice-Hall, 1974.

[35] L. Dobrjanskyj and F. Freudenstein. Some applications of graph theory to the structural analysis of mechanisms. *ASME Journal of Engineering for Industry*, 89:153–158, 1967.

[36] K.V.D. Doel and D.K. Pai. Performance measures for robot manipulators: a unified approach. *International Journal of Robotics Research*, 15(1):92–111, 1996.

[37] J. Duffy. *Analysis of mechanisms and robot manipualtors.* John Wiley & Sons, 1980.

[38] J. Duffy and C. Crane. A displacement analysis of the general spacial $7r$ mechanism. *Mechanism and Machine Theory*, 15:153–169, 1980.

[39] R. Featherstone. *Robot Dynamics Algorithms.* Kluwer Academic Publishers, 1987.

[40] R. Featherstone. Explicit modeling of general task spaces for inverse kinematics. In *Advances in Robot Kinematics and Computational Geometry*, pages 301–308. Cambridge, MIT Press, 1994.

[41] F. Freudenstein and E. Primrose. On the analysis and sythesis of the workspace of a three-link, turning pair connected robot arm. *ASME Journal of Mechanisms, Transmission, and Automation in Design*, 106:365–370, 1984.

[42] T. Fukuda and S. Nakagawa. Dynamically reconfigurable robotic system. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1581–1586, Philadelphia, PA, April 1988.

[43] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss. Structure decision method for self organizing robots based on cell structures-cebot. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 695–700, Scottsdale, AZ, 1989.

[44] T. Fukuda, T. Ueyama, and F. Arai. Control strategy for a network of cellular robots. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1616–1621, Sacramento, CA, April 1991.

[45] A. A. Goldenberg and D. L. Lawrence. A generalized solution to the inverse kinematics of robotic manipulators. *ASME Journal of Dynamic Systems, Measurement, and Control*, 107:103–106, 1985.

[46] C. Gosselin and J. Angeles. A global performance index for the kinematic optimization of robotic manipulators. *ASME Journal of Mechanical Design*, 113:220–226, 1991.

[47] K.C. Gupta. On the nature of robot workspace. *International Journal of Robotics Research*, 5(2):112–121, 1986.

[48] K.C. Gupta. Kinematic analysis of manipulator using the zero reference position description. In J.M. McCarthy, editor, *The Kinematics of Robot Manipulators*, pages 3–11, Cambridge, MA, 1987. MIT Press.

[49] K.C. Gupta and K. Kazerounian. Improved numerical solution of inverse kinematics of robots. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 743–748, Philadelphia, PA, April 1988.

[50] K.C. Gupta and B. Roth. Design considerations for manipulator workspace. *ASME Journal of Mechanical Design*, 104:704–711, 1982.

[51] J. Han, W.K. Chung, and S.H. Kim. Task based design of modular robot manipulator using efficient genetic algorithm. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 507–511, New Mexico, USA, 1997.

[52] F. Harary and H. Yan. Logical foundations of kinematic chains: graphs, line graphs, and hypergraphs. *ASME Journal of Mechanical Design*, 112:79–83, March 1990.

[53] S.A. Hayati. Robot arm geometric parameter estimation. In *Proceedings of IEEE Conference on Decision and Control*, pages 1477–1483, 1983.

[54] S.A. Hayati and M. Mirmirani. Improving the absolute positioning accuracy of robot manipulators. *Journal of Robotic Systems*, 112(2):397–413, 1985.

[55] S.A. Hayati and G.P. Roston. Inverse kinematic solution for near-simple robots and its application to robot calibration. In *Recent Trends in Robotics: Modeling, Control, and Education*, pages 41–50. Elsevier Science, 1986.

[56] S.A. Hayati and G.P. Roston. Robot geometry calibration. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 947–951, Philadelphia, PA, April 1988.

[57] J.M. Hollerbach. A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transaction on Systems, Man, and Cybernetics*, 10:730–736, November 1980.

[58] J.M. Hollerbach. *Optimum kinematics design of a seven degree of freedom manipulator*. MIT Press, Cambridge, MA, 1985.

[59] J.M. Hollerbach. A survey of kinematic calibration. In *The Robotics Review I*, pages 207–242. Cambridge, MIT Press, 1989.

[60] J.M. Hollerbach and D.J. Bernett. Automatic kinematic calibration using a motion tracking system. In *Modeling and Control of Robotic Manipulators and Manufacturing Processes*, pages 93–100, 1987.

[61] R.L. Huston, C.E. Passerrello, and M.W. Harlow. Dynamics of multirigid-body system. *ASME Journal of Applied Mechanics*, 45(4):889–894, 1978.

[62] R.P. Judd and A.B. Knasinski. A technique to calibrate industrial robots with experimental verification. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 351–357, Raleigh, NC, March 1987.

[63] T.R. Kane and D.A. Levinson. The use of kane's dynamic equation in robotics. *International Journal of Robotics Research*, 2(3):1–21, 1983.

[64] K. Kazerounian and G.Z. Qian. Kinematic calibration of robotics manipulators. In *ASME Design Technology Conferences : Biennial Mechanisms Conference*, pages 261–266, 1988.

[65] K. Kazerounian and G.Z. Qian. Kinematic calibration of robotic manipulator. *ASME Journal of Mechanisms, Transmission, and Automation in Design*, 111:482–487, 1989.

[66] L. Kelmar and P. Khosla. Automatic generation of kinematics for a reconfigurable modular manipulator system. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 663–668, Philadelphia, PA, April 1988.

[67] P.K. Khosla, C.P. Neuman, and F.B. Prinz. An algorithm for seam tracking applications. *International Journal of Robotics Research*, 4(1):27–41, 1985.

[68] J.O. Kim and P. Khosla. Design of space shuttle tile servicing robot: an application of task based kinematic design. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 867–874, Altanta, GA, May 1993.

[69] J.O. Kim and P. Khosla. A formulation for task-based design of robot manipulators. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2310–2317, 1993.

[70] C.A. Klein and B.E. Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *International Journal of Robotics Research*, 6(2):72–83, 1987.

[71] A. Kumar and K.J. Waldron. The workspaces of a mechanical manipulator. *ASME Journal of Mechanical Design*, 103:665–672, 1981.

[72] C.S.G. Lee. Robot arm kinematics, dynamics, and control. *IEEE Computer*, 15(12):62–68, 1982.

[73] C.S.G. Lee, B.H. Lee, and R. Nigam. Development of the generalized d'alembert equation of motion for mechanical design. In *Proceedings of IEEE Conference on Decision and Control*, pages 9998–9999, 1983.

[74] H.Y. Lee and C.G. Liang. Displacement analysis of the general serial 7-link 7-r mechanism. *Mechanism and Machine Theory*, 23:219–226, 1988.

[75] H.Y. Lee and C.G. Liang. A new vector theory for the analysis of spatial mechanisms. *Mechanism and Machine Theory*, 23:209–217, 1988.

[76] H.Y. Lee and C.F. Reinholtz. Inverse kinematics of serial-chain manipulators. *ASME Journal of Mechanical Design*, 118:396–404, 1996.

[77] G. Legnani and R. Riva. Kinematics of modular robots. In *Proceedings of World Congress on the Theory of Machines and Mechanisms*, pages 1159–1162, 1987.

[78] J. Lenarčič. An efficient numerical approach for calculating the inverse kinematics for robot manipulators. *Robotica*, pages 21–26, 1985.

[79] Z. Li. Geometrical considerations of robot kinematics. *IEEE Transactions on Robotics and Automation*, 5(3):139–145, 1990.

[80] W.C. Lin, J.B. Ross, and M. Ziegler. Semiautomatic calibration of robot manipulator for visual inspection task. *Journal of Robotic Systems*, 3:19–39, 1986.

[81] J.Y.S. Luh, M.W. Walker, and R.P. Pual. On-line computational scheme for mechanical manipulators. *ASME Journal of Dynamic Systems, Measurement, and Control*, 102:103–110, 1980.

[82] D. Manocha and J. F. Canny. Real time inverse of the general $6r$ manipulators. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 383–389, Nice France, May 1992.

[83] R. Manseur and K.L. Doty. A fast algorithm for inverse kinematic analysis of robot manipulators. *International Journal of Robotics Research*, 4(2):21–37, 1988.

[84] T. Matsumaru. Design and control of the modular robot system: Tomms. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 2125–2131, Nagoya, Japan, May 1995.

[85] D. R. Meldrum, G. Rodriguez, and G. F. Franklin. An order(n) recursive inversion of the jacobian for an n-link serial manipulator. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1175–1180, Sacramento, CA, April 1991.

[86] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 2 edition, 1994.

[87] B.W. Mooring. The effect of joint axis misalignment on robot positioning accuracy. In *Proceedings of Computers in Engineering Conference and Exhibit*, pages 151–155, 1983.

[88] B.W. Mooring. An improved method for identifying the kinematic parameters in a six axis robot. In *Proceedings of Computers in Engineering Conference and Exhibit*, pages 79–84, 1984.

[89] B.W. Mooring, Z.S. Roth, and M.R. Driels. *Fundamentals of Manipulator Calibration.* John Wiley and Sons, 1991.

[90] R. Murray, Z. Li, and S.S. Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.

[91] K. Okamura and F.C. Park. Kinematic calibration and the product of exponential formula. *Robotica*, 14:415–421, 1996.

[92] B. Paden and S. Sastry. Optimal kinematic design of 6r manipulators. *International Journal of Robotics Research*, 7(2):43–61, 1988.

[93] C.J.J. Paredis. *An Agent-Based Approach to the Design of Rapidly Deployable Fault Tolerant Manipulators.* PhD thesis, Carnegie Mellon University, USA, 1996.

[94] C.J.J. Paredis, H.B. Brown, R.W. Casciola, J.E. Moody, and P.K. Khosla. A rapidly deployable manipulator system. In *International Workshop on Some Critical Issues in Robotics*, pages 175–185, Singapore, 1995.

[95] C.J.J. Paredis and P. Khosla. Kinematic design of serial link manipulators from task specifications. *International Journal of Robotics Research*, 12(3):274–287, 1993.

[96] C.J.J. Paredis and P.K. Khosla. Synthesis methodology for task based reconfiguration of modular manipulator systems. In *International Symposium on Robotics Research*, Hidden Valley, PA, 1993.

[97] C.J.J. Paredis and P.K. Khosla. Design of modular fault tolerant manipulators. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 371–383. A.K. Peters, 1995.

[98] C.J.J. Paredis and P.K. Khosla. Agent-based design of fault tolerant manipulators for satellite docking. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 3473–3480, New Mexico, USA, 1997.

[99] F.C. Park. On the optimal kinematic design of spherical and spatial mechanisms. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1530–1535, Sacramento, CA, April 1991.

[100] F.C. Park. Computational aspect of manipulators via product of exponential formula for robot kinematics. *IEEE Transactions on Automatic Control*, 39(9):643–647, 1994.

[101] F.C. Park. Optimal robot design and differential geometry. *ASME Special 50th Anniversary Design Issue*, 117:87–91, 1995.

[102] F.C. Park. A coordinate-free description of robot dynamics. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 2282–2287, New Mexico, USA, 1997.

[103] F.C. Park and J.E. Bobrow. A recursive algorithm for robot dynamics using lie groups. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1535–1540, San Diego, CA, June 1994.

[104] F.C. Park, J.E. Bobrow, and S.R. Ploen. A lie group formulation of robot dynamics. *International Journal of Robotics Research*, 14(6):609–618, December 1995.

[105] F.C. Park and R.W. Brockett. Kinematic dexterity of robotic mechanisms. *International Journal of Robotics Research*, 13(1):1–15, 1994.

[106] F.C. Park and K. Okamura. Kinematic calibration and the product of exponential formula. In *Advances in Robot Kinematics and Computational Geometry*, pages 119–128. Cambridge, MIT Press, 1994.

[107] R. P. Paul, B. Shimano, and G. E. Mayer. Kinematic control equation for simple manipulators. *IEEE Transaction on Systems, Man, and Cybernetics*, 11(6):80–86, 1981.

[108] R.P. Paul. *Robot Manipulators: Mathematics, Programming, and Control*. MIT Press, Cambridge, MA, 1981.

[109] S.R. Ploen and F.C. Park. Coordinate-invariant algorithms for robot dynamics. In *International Workshop on Some Critical Issues in Robotics*, pages 115–128, Singapore, 1995.

[110] M. Raghavan and B. Roth. Kinematic analysis of the $6R$ manipulator of general gepmetry. In *International Symposium on Robotics Research*, pages 314–320, 1989.

[111] B. Roth. Performance evaluation of manipulators from a kinematic point of view. In *NBS Special Publication 495*, pages 39–61. National Bureau of Standards, 1976.

[112] Z. Roth, B. W. Mooring, and B. Ravani. An overview of robot calibration. *IEEE Journal on Robotics and Automation*, 3(5):377–386, 1987.

[113] Jeffrey Russakow, Oussama Khatib, and Stephen M. Rock. Extended operational space formulation for serial-to-parallel chain (branching) manipulators. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1056–1061, Nagoya, Japan, May 1995.

[114] J.K. Salisbury and J.J. Craig. Articulated hands: Force control and kinematic issues. *International Journal of Robotics Research*, 1(1):4–17, 1982.

[115] D. Schmitz, P. Khosla, and T. Kanade. The cmu reconfigurable modular manipulator system. Technical Report CMU-RI-TR-88-7, Carnegie Mellon University, 1988.

[116] S. Skiena. *Implementing Discrete Mathematics.* Addison-Wesley, Redwood City, CA, 1990.

[117] M.W. Spong. Remarks on robot dynamics: canonical transformation and Riemannian geometry. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 554–559, Nice France, May 1992.

[118] W. Stephen. *Mathematica: A System for Doing Mathematics by Computers.* Addison-Wesley, USA, 2 edition, 1991.

[119] H.W. Stone. *Kinematic Modeling, Identification, and Control of robotic Manipulators.* Kluwer Academic Publ., 1987.

[120] K. Sugimoto and T. Okada. Compensation of positioning errors caused by geometric deviations in robot system. In H. Hanafusa and H. Inoue, editors, *International Symposium on Robotics Research*, pages 231–236. MIT Press, Cambridge, MA, 1985.

[121] C. H. Suh and C. W. Radcliffe. *Kinematics and Mechanisms Design.* John Wiley & Sons, 1978.

[122] C.T. Tan, G. Yang, and I.M. Chen. A modular kinematic calibration algorithm for industrial robots: simulation and experiment. In *IASTED International Conference on Robotics and Manufacturing*, Canada, pages 231–234, 1998. submitted for publication.

[123] C.T. Tan, G. Yang, and I.M. Chen. A generic kinematic calibration model for the open chain robots. In *Proceedings of International Conference on Control, Automation, Robotics, and Vision*, pages 1503-1507, Singapore, 1998.

[124] D. Tesar and M.S. Butler. A generalized modular architecture for robot structures. *ASME Journal of Manufacturing Review*, 2(2):91–117, 1989.

[125] L.-W. Tsai and A.P. Morgan. Solving the kinematics of the most general 6 and 5 dof manipulators by continuation methods. *ASME Journal of Mechanisms, Transmission, and Automation in Design*, 107:189–200, 1985.

[126] W.K. Veitschegger and C.H. Wu. Robot accuracy analysis based on kinematics. *IEEE Transactions on Robotics and Automation*, 2:171–179, 1986.

[127] C. Wampler and A.P. Morgen. Solving the $6r$ inverse position problem using a geometric-case solution methodology. *Mechanism and Machine Theory*, 26:91–106, 1991.

[128] D.E. Whitney, C.A. Lozinski, and J.M. Rourke. Industrial robot forward calibration method and results. *Journal of Dynamic Systems*, 108:1–8, 1986.

[129] R. J. Wilson and J. J. Watkins. *Graphs - An Introductory Approach.* John Wiley & Sons Inc., New York, USA, 1989.

[130] L.S. Woo. Type synthesis of plane linkages. *ASME Journal of Engineering for Industry*, 89:159–172, 1967.

[131] C. Wu. The kinematic error model for the design of robot manipulators. In *Proceedings of the American Control Conference*, pages 497–502, 1983.

[132] K.H. Wurst. The conception and construction of a modular robot system. In *International Symposium on Industrial Robotics*, pages 37–44, 1986.

[133] H.-S. Yan and Y.-W. Hwang. The specialization of mechanisms. *Mechanism and Machine Theory*, 26(6):541–551, 1991.

[134] D.C.H. Yang and T.W. Lee. On the workspace of mechanical manipulator. *ASME Journal of Mechanisms, Transmission, and Automation in Design*, 105:62–69, 1983.

[135] F.C. Yang and E.J. Hang. Numerical analysis of the kinematic working capability of mechanisms. *ASME Journal of Mechanical Design*, 116:111–118, 1994.

[136] G. Yang and I.M. Chen. Kinematic calibration of modular reconfigurable robots. In *Proceedings of International Conference on Control, Automation, Robotics, and Vision*, pages 1845–1849, Singapore, 1996.

[137] G. Yang and I.M. Chen. A novel kinematic calibration algorithm for modular reconfigurable robotic systems. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 3197–3202, 1997.

[138] G. Yang and I.M. Chen. Reduced DOF modular robot configurations. In *Proceedings of International Conference on Control, Automation, Robotics, and Vision*, pages 1513–1517, Singapore, 1998.

[139] G. Yang and I.M. Chen. Task-based optimization of modular robot configurations - MDOF approach. *Mechanism and Machine Theory*, 1999 (to appear).

[140] T.C. Yih and Y. Youm. Matrix solution for the inverse kinematics of robots. In *ASME International Conference*, pages 371–375, 1990.

[141] T. Yoshikawa. Manipulability of robotic mechanisms. *International Journal of Robotics Research*, 4(2):3–9, 1985.

[142] H. Zhuang and Z.S. Roth. Robot calibration using the cpc error model. *Journal of Robotics and Computer-Integrated Manufacturing*, 9(3):227–237, 1992.

[143] H. Zhuang and Z.S. Roth. A linear solution to the kinematic parameter identification of robot manipulator. *IEEE Transactions on Robotics and Automation*, 9(2):174–185, 1993.

[144] H. Zhuang, Z.S. Roth, and H. Fumio. A complete and parametrically continuous kinematic model for robot manipulators. *IEEE Transactions on Robotics and Automation*, 8(4):451–463, 1992.