

Introduction to Robotics Analysis, Systems, Applications

Saeed B. Niku

*Mechanical Engineering Department
California Polytechnic State University
San Luis Obispo*



Prentice Hall
Upper Saddle River, NJ 07458

Library of Congress Cataloging-in-Publication Data

CIP DATA ON FILE

Vice President and Editorial Director, ECS: *Marcia Horton*
Acquisitions Editor: *Eric Frank*
Editorial Assistant: *Jessica Romeo*
Vice President of Production and Manufacturing, ESM: *David W. Riccardi*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *David A. George*
Production Editor: *Lakshmi Balasubramanian*
Director of Creative Services: *Paul Belfanti*
Art Director: *Jayne Conte*
Cover Designer: *Bruce Kenselaar*
Art Editor: *Adam Velthaus*
Manufacturing Manager: *Trudy Pisciotti*
Manufacturing Buyer: *Pat Brown*
Marketing Manager: *Holly Stark*



© 2001 Prentice Hall
Prentice Hall, Inc.
Upper Saddle River, New Jersey 07458

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4

ISBN 0-13-061309-6

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Pearson Education Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Preface

As one of my students once said years ago, “in the life of every product, there comes a time when you have to shoot the designer and go into production.” It seems that the same is true for a book. An author of a textbook such as this one may go on forever trying to cover any and every conceivable subject related to the book in order to generate an all-encompassing book that satisfies every teacher and student. But the intention behind writing this book was not that at all. The intention was to write a book that has most subjects that an undergraduate engineering student or a practicing engineer may need to know to be familiar with the subject, to be able to understand robots and robotics, to be able to design a robot, and to be able to integrate a robot in appropriate applications. As such, it covers all necessary fundamentals of robotics, its components and subsystems, and its applications.

This book was originally written for Cal Poly Mechanical Engineering Department’s Robotics course. With encouragement from different people, it was somewhat modified to the present form. 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, it also covers microprocessor applications, vision systems, sensors, and electric motors. Thus, it can easily be used by mechanical engineers, electronic and electrical engineers, computer engineers, and engineering technologists.

The book comprises nine chapters. Chapter 1 covers introductory subjects that familiarize the reader with the necessary background information that is used in the rest of the book. This includes some historical information, robot components, robot characteristics, robot languages, and robotic applications. Chapter 2 covers the forward and inverse kinematics of robots, including frame representations, transformations, position and orientation analysis, and the Denavit–Hartenberg representation of robot kinematics. Chapter 3 continues with

- Introductory material and review: 3 lectures
- Kinematics of position: 9 lectures
- Differential motions: 5 lectures
- Robot dynamics and force control: 5 lectures
- Path and trajectory planning: 4 lectures
- Actuators: 4 lectures
- Sensors: 3 lectures
- Vision systems: 5 lectures
- Fuzzy logic: 2 lectures
- Exams and review: 2 lectures

The book also features a design project that starts in Chapter 2 and continues throughout the book. At the end of each chapter, the student is directed to continue with the design project as it relates to the present chapter. Thus, by the end of the book, a complete robot is designed. In addition, a rolling-cylinder robot rover design project is also introduced in Chapter 6 and continues in Chapter 7.

I would like to thank all the people who, in one way or another, have contributed to this book. This includes my colleagues in the mechanical engineering department and the university who provided me with a sabbatical to write the first draft, all the countless individuals who did the research, development, and the hard work that came before my time and that enabled me to learn the subject myself, all the students and anonymous reviewers who made countless suggestions to improve the first draft, my editors Eric Frank and Lakshmi Balasubramanian, all the staff at Prentice Hall, who worked diligently to get a professional book out on time, and, of course, my family, who let me work on this manuscript for long hours instead of spending the time with them. 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

differential motions and velocity analysis of robots and frames. Chapter 4 presents an analysis of robot dynamics and forces. Lagrangian mechanics is used as the primary method of analysis and development for this chapter. Chapter 5 discusses methods of path and trajectory planning, both in joint-space and in Cartesian-space. Chapter 6 covers actuators, including hydraulic devices, electric motors such as DC servomotors and stepper motors, Pneumatic devices, as well as many other novel actuators. It also covers microprocessor control of these actuators. Although this book 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 7 is a discussion of sensors that are used in robotics and robotic applications. Chapter 8 covers vision systems, including many different techniques for image processing and image analysis. Chapter 9 cover some 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 instead an introduction to it. 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.

Since the book is written for senior-level engineering students or for practicing engineers, the assumption is that the users are familiar with matrix algebra, as well as with basic feedback control theory and analysis. For this reason, except for some basic review, this material is not separately covered in this book. Obviously, to know enough control theory to be proficient in it, one has to have access to a complete controls book, something that is beyond the scope of a robotics book.

Most of the material in this book is generally covered in a four-unit, 10-week-long course at Cal Poly, with three one-hour lectures and one three-hour lab. However, it is easily possible to cover the entire course in a semester-long course as well. 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: 3 lectures
- Kinematics of position: 7 lectures
- Differential motions: 4 lectures
- Robot dynamics and force control: 2 lectures
- Path and trajectory planning: 1 lecture
- Actuators: 3 lectures
- Sensors: 3 lectures
- Vision systems: 4 lectures
- Fuzzy logic: 1 lectures
- Exam and review: 2 lectures

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

*Dedicated to
Shohreh, Adam, and Alan Niku
and to
Sara Niku and memory of Saleh Niku*

Contents

1 Fundamentals

- | | | |
|-------|--|----|
| 1.1. | Introduction | 1 |
| 1.2. | What is a Robot? | 2 |
| 1.3. | Classification of Robots | 2 |
| 1.4. | What is Robotics? | 4 |
| 1.5. | History of Robotics | 4 |
| 1.6. | Advantages and Disadvantages of Robots | 5 |
| 1.7. | Robot Components | 6 |
| 1.8. | Robot Degrees of Freedom | 8 |
| 1.9. | Robot Joints | 11 |
| 1.10. | Robot Coordinates | 11 |
| 1.11. | Robot Reference Frames | 12 |
| 1.12. | Programming Modes | 13 |
| 1.13. | Robot Characteristics | 15 |
| 1.14. | Robot Workspace | 16 |
| 1.15. | Robot Languages | 16 |
| 1.16. | Robot Applications | 20 |
| 1.17. | Other Robots and Applications | 24 |

1.18. Social Issues	25
1.19. Summary	25
References	26
Problems	27
2 Robot Kinematics: Position Analysis	29
2.1. Introduction	29
2.2. Robots as Mechanisms	29
2.3. Matrix Representation	31
2.3.1. <i>Representation of a point in space</i>	31
2.3.2. <i>Representation of a vector in space</i>	32
2.3.3. <i>Representation of a frame at the origin of a reference frame</i>	33
2.3.4. <i>Representation of a frame in a reference frame</i>	34
2.3.5. <i>Representation of a Rigid Body</i>	35
2.4. Homogeneous Transformation Matrices	38
2.5. Representation of Transformations	38
2.5.1. <i>Representation of a pure translation</i>	39
2.5.2. <i>Representation of a pure rotation about an axis</i>	40
2.5.3. <i>Representation of combined transformations</i>	43
2.5.4. <i>Transformations relative to the rotating</i>	46
2.6. Inverse of Transformation Matrices	48
2.7. Forward and Inverse Kinematics of Robots	53
2.7.1. <i>Forward and Inverse Kinematic Equations for Position</i>	54
2.7.2. <i>Forward and Inverse Kinematic Equations for Orientation</i>	59
2.7.3. <i>Forward and Inverse Kinematic Equations for Position and Orientation</i>	67
2.8. Denavit–Hartenberg Representation of Forward Kinematic Equations of Robots	67
2.9. The Inverse Kinematic Solution of Robots	76
2.10. Inverse Kinematic Programming of Robots	80
2.11. Degeneracy and Dexterity	82
2.12. The Fundamental Problem with the Denavit–Hartenberg Representation	83
2.13. Design Project 1: A three-degree-of-freedom Robot	85
2.14. Summary	86
References	87
Problems	88

3 Differential Motions and Velocities	95
3.1. Introduction	95
3.2. Differential Relationships	95
3.3. Jacobian	97
3.4. Differential Motions of a Frame	99
3.4.1. Differential Translations	100
3.4.2. Differential Rotations	100
3.4.3. Differential Rotation about a general axis \hat{k}	101
3.4.4. Differential Transformations of a Frame	102
3.5. Interpretation of the Differential Change	104
3.6. Differential Changes Between Frames	104
3.7. Differential Motions of a Robot and Its Hand Frame	106
3.8. Calculation of the Jacobian	107
3.9. How to Relate the Jacobian and the Differential Operator	110
3.10. Inverse Jacobian	111
3.11. Design Project	115
3.12. Summary	116
References	116
Problems	117
4 Dynamic Analysis and Forces	119
4.1. Introduction	119
4.2. Lagrangian Mechanics: A Short Overview	120
4.3. Effective Moments of Inertia	127
4.4. Dynamic Equations for Multiple-Degree-of-Freedom Robots	128
4.4.1. Kinetic Energy	128
4.4.2. Potential Energy	132
4.4.3. The Lagrangian	133
4.4.4. Robot's Equations of Motion	133
4.5. Static Force Analysis of Robots	139
4.6. Transformation of Forces and Moments Between Coordinate Frames	141
4.7. Design Project	143
4.8. Summary	143

References	144
Problems	144
5 Trajectory Planning	147
5.1. Introduction	147
5.2. Path vs. Trajectory	147
5.3. Joint-Space vs. Cartesian-Space Descriptions	148
5.4. Basics of Trajectory Planning	150
5.5. Joint-Space Trajectory Planning	153
5.5.1. <i>Third-Order Polynomial Trajectory Planning</i>	154
5.5.2. <i>Fifth-Order Polynomial Trajectory Planning</i>	157
5.5.3. <i>Linear Segments with Parabolic Blends</i>	157
5.5.4. <i>Linear Segments with Parabolic Blends and Via Points</i>	160
5.5.5. <i>Higher Order Trajectories</i>	161
5.5.6. <i>Other Trajectories</i>	165
5.6. Cartesian-Space Trajectories	165
5.7. Continuous Trajectory Recording	170
5.8. Design Project	170
5.9. Summary	171
References	171
Problems	172
6 Actuators	173
6.1. Introduction	173
6.2. Characteristics of Actuating Systems	174
6.2.1. <i>Weight, Power-to-Weight Ratio, Operating Pressure</i>	174
6.2.2. <i>Stiffness vs. Compliance</i>	174
6.2.3. <i>Use of Reduction Gears</i>	175
6.3. Comparison of Actuating Systems	178
6.4. Hydraulic Devices	178
6.5. Pneumatic Devices	184
6.6. Electric Motors	186
6.6.1. <i>DC Motors</i>	188
6.6.2. <i>AC Motors</i>	189
6.6.3. <i>Brushless DC motors</i>	189
6.6.4. <i>Direct Drive Electric Motors</i>	189

6.6.5. Servomotors	190
6.6.6. Stepper Motors	191
6.7. Microprocessor Control of Electric Motors	207
6.7.1. Pulse Width Modulation.	209
6.7.2. Direction Control of DC Motors with an H-Bridge	210
6.8. Magnetostrictive Actuators	210
6.9. Shape-Memory Type Metals	211
6.10. Speed Reduction	212
6.11. Design Project 1	215
6.12. Design Project 2	215
6.13. Summary	216
References	217
Problems	218

7 Sensors	219
------------------	------------

7.1. Introduction	219
7.2. Sensor Characteristics	219
7.3. Position Sensors	222
7.3.1. Potentiometers	222
7.3.2. Encoders	223
7.3.3. Linear Variable Differential Transformers (LVDT)	226
7.3.4. Resolvers	228
7.3.5. Time-of-Travel Displacement Sensor	229
7.4. Velocity Sensors	229
7.4.1. Encoders	230
7.4.2. Tachometers	230
7.4.3. Differentiation of position signal	230
7.5. Acceleration Sensors	230
7.6. Force and Pressure Sensors	231
7.6.1. Piezoelectric	231
7.6.2. Force Sensing resistor	231
7.6.3. Strain gauges	231
7.7. Torque Sensors	233
7.8. Microswitches	233
7.9. Light and Infrared Sensors	233
7.10. Touch and Tactile Sensors	234

- 7.11. Proximity Sensors 236
 - 7.11.1. *Magnetic Proximity Sensors* 236
 - 7.11.2. *Optical Proximity Sensors* 236
 - 7.11.3. *Ultrasonic Proximity Sensors* 237
 - 7.11.4. *Inductive Proximity Sensors* 237
 - 7.11.5. *Capacitive Proximity Sensors* 237
 - 7.11.6. *Eddy Current Proximity Sensors* 238
- 7.12. Range-finders 238
 - 7.12.1. *Ultrasonic Range Finders* 239
 - 7.12.2. *Light Based Range Finders* 240
- 7.13. Sniff Sensors 241
- 7.14. Vision Systems 241
- 7.15. Voice Recognition Devices 241
- 7.16. Voice Synthesizers 242
- 7.17. Remote Center Compliance (RCC) Device 242
- 7.18. Design Project 245
- 7.19. Summary 246
- References 246

8 Image Processing and Analysis with Vision Systems

248

- 8.1. Introduction 248
- 8.2. Image Processing versus Image Analysis 248
- 8.3. Two- and Three-Dimensional Image Types 249
- 8.4. What is an Image 249
- 8.5. Acquisition of Images 250
 - 8.5.1. *Vidicon Camera* 250
 - 8.5.2. *Digital Camera* 252
- 8.6. Digital Images 254
- 8.7. Frequency Domain vs. Spatial Domain 254
- 8.8. Fourier Transform of a Signal and its Frequency Content 255
- 8.9. Frequency Content of an Image; Noise, Edges 257
- 8.10. Spatial Domain Operations: Convolution Mask 259
- 8.11. Sampling and Quantization 262
- 8.12. Sampling Theorem 263
- 8.13. Image-Processing Techniques 267

- 8.14. Histogram of Images 267
- 8.15. Thresholding 268
- 8.16. Connectivity 269
- 8.17. Noise Reduction 271
 - 8.17.1 *Convolution Masks* 272
 - 8.17.2. *Image Averaging* 273
 - 8.17.3. *Frequency Domain* 274
 - 8.17.4. *Median Filters* 274
- 8.18. Edge Detection 275
- 8.19. Hough Transform 279
- 8.20. Segmentation 282
- 8.21. Segmentation by Region Growing and Region Splitting 282
- 8.22. Binary Morphology Operations 284
 - 8.22.1. *Thickening Operation* 284
 - 8.22.2. *Dilation* 285
 - 8.22.3. *Erosion* 285
 - 8.22.4. *Skeletonization* 286
 - 8.22.5. *Open Operation* 287
 - 8.22.6. *Close Operation* 287
 - 8.22.7. *Fill Operation* 287
- 8.23. Gray Morphology Operations 288
 - 8.23.1. *Erosion* 288
 - 8.23.2. *Dilation* 288
- 8.24. Image Analysis 288
- 8.25. Object Recognition by Features 288
 - 8.25.1. *Basic Features Used for Object Identification* 289
 - 8.25.2. *Moments* 290
 - 8.25.3. *Template Matching* 297
 - 8.25.4. *Discrete Fourier Descriptors* 297
 - 8.25.5. *Computed Tomography* 297
- 8.26. Depth Measurement with Vision Systems 298
 - 8.26.1. *Scene Analysis vs. Mapping* 298
 - 8.26.2. *Range Detection and Depth Analysis* 299
 - 8.26.3. *Stereo Imaging* 299
 - 8.26.4. *Scene Analysis with Shading and Sizes* 300
- 8.27. Specialized Lighting 301
- 8.28. Image Data Compression 302
 - 8.28.1. *Intraframe Spatial Domain Techniques* 302
 - 8.28.2. *Interframe Coding* 303
- 8.29. Real-Time Image Processing 304

8.30. Heuristics	304
8.31. Applications of Vision Systems	305
8.32. Design project	306
8.33. Summary	306
References	307
Problems	308
9 Fuzzy Logic Control	311
9.1. Introduction	311
9.2. Fuzzy Control: What is needed	313
9.3. Crisp Values vs. Fuzzy Values	314
9.4. Fuzzy Sets: Degrees of Membership and Truth	314
9.5. Fuzzification	315
9.6. Fuzzy Inference Rule Base	316
9.7. Defuzzification	318
9.7.1. <i>Center-of-Gravity Method</i>	318
9.7.2. <i>Mamdani's Inference Method</i>	318
9.8. Simulation of Fuzzy Logic Controller	322
9.9. Applications of Fuzzy Logic in Robotics	323
9.10. Design Project	327
9.11. Summary	328
References	328
Problems	328
APPENDIX A	331
A.1. Matrix Algebra and Notation: A Review	331
A.2. Calculation of an Angle From its Sine, Cosine, or Tangent	336
Problems	338
INDEX	339

Fundamentals

1.1 INTRODUCTION

Robotics, in different forms, has always been on people's minds, since the time we first built things. You may have seen machines that artisans made that tried to mimic a human's motions and behavior. Two examples of such machines are the statues in Venice that hit the clock on the hour and toys with repeating movements. Hollywood and movies have taken this desire one step further by portraying robots and humanoids as even superior to humans.

Although in principle humanoids are robots and are designed and governed by the same basics, in this book, we will primarily study industrial manipulator type robots. This book covers some basic introductory material that will get you familiar with the subject, presents analyses of the mechanics of robots, including kinematics and dynamics of robots, and discusses the elements that are used in robots and in robotics, such as actuators, sensors, and vision systems.

Robots are very powerful elements of today's industry. They are capable of performing many different tasks and operations precisely and do not require common safety and comfort elements humans need. However, it takes much effort and many resources to make a robot function properly. Most companies that made robots in the mid-1980s no longer exist, and only companies that made industrial robots remain in the market (such as Adept Robotics, Staubli Robotics, and Fanuc Robotics, North America, Inc.). Early predictions about the possible number of robots in industry never materialized, because high expectations could not be satisfied with the present robots. As a result, although there are many thousands of robots in industry, they have not overwhelmingly replaced workers. They are used where they are useful. As with humans, robots can do certain things, but not other things. As long as they are designed properly for the intended purpose, they are very useful and will continue to be used.

2 Chapter 1 Fundamentals

The subject of robotics covers many different areas. Robots alone are hardly ever useful. They are used together with other devices, peripherals, 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?

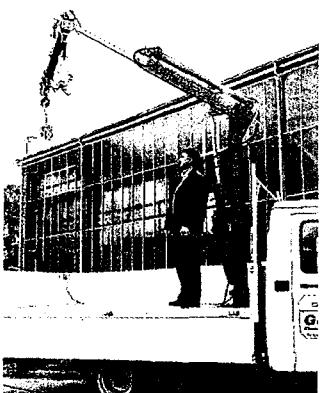
If you compare a conventional robotic manipulator with a crane attached to, say, a utility or towing vehicle, you will notice that the robot manipulator is very similar to the crane. Both possess a number of links attached serially to each other with joints, where each joint can be moved by some type of actuator. In both systems, the “hand” of the manipulator can be moved in space and be placed in any desired location within the workspace of the system, each one can carry a certain amount of load, and each one is controlled by a central controller which controls the actuators. However, one is called a robot and the other a manipulator (or, in this case, a crane). The fundamental difference between the two is that the crane is controlled by a human who operates and controls the actuators, whereas the robot manipulator is controlled by a computer that runs a program. This difference between the two 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 that is under the supervision of the computer, which, itself, is running some type of a program. Thus, if the program is changed, the actions of the robot will be changed accordingly. The intention is to have a device that can perform many different tasks and thus is very flexible in what it can do, without having to redesign the device. Thus, the robot is designed to be able to perform any task that can be programmed (within limit, of course) 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 to be a robot. By American standards, a device must be easily reprogrammable to be considered a robot. Thus, manual-handling devices (i.e., a device that has multiple degrees of freedom and is actuated by an operator) or fixed-sequence robots (i.e., any device controlled by hard stops to control actuator motions on a fixed sequence and difficult to change) are not considered to be robots.

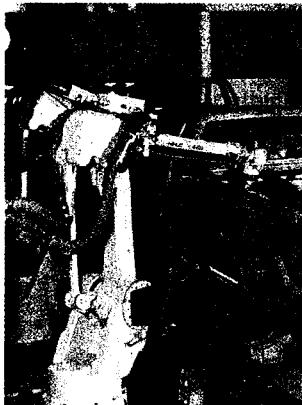
1.3 CLASSIFICATION OF ROBOTS

The following is the classification of robots according to the Japanese Industrial Robot Association (JIRA):

- Class 1: *Manual-Handling Device*: A device with multiple degrees of freedom that is actuated by an operator.



(a)



(b)

Figure 1.1 A robot and a crane are very similar in the way they operate and in the way they are designed. However, the crane is controlled by an operator, whereas the robot is controlled by a computer. Thus, by simply changing the computer program, the robot will function differently. (a) A Kuhnezug truck-mounted crane. Reprinted with permission from Kuhne zug Fordertechnik GmbH. (b) Fanuc S-500 robots performing seam-sealing on a truck. Reprinted with permission from Fanuc Robotics, North America, Inc.

- Class 2: *Fixed-Sequence Robot*: A device that performs the successive stages of a task according to a predetermined, unchanging method and is hard to modify.
- Class 3: *Variable-Sequence Robot*: Same as class 2, but easy to modify.
- Class 4: *Playback Robot*: A human operator performs the task manually by leading the robot, which records the motions for later playback. The robot repeats the same motions according to the recorded information.
- Class 5: *Numerical Control Robot*: The operator supplies the robot with a movement program rather than teaching it the task manually.
- Class 6: *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.

The Robotics Institute of America (RIA) only considers classes 3–6 as robots. The Association Française de Robotique (AFR) has the following classification:

- Type A: Handling devices with manual control to telerobotics.
- Type B: Automatic handling devices with predetermined cycles.
- Type C: Programmable, servo controlled robots with continuous or point-to-point trajectories.
- Type D: Same as type C, but with capability to acquire information from its environment.

1.4 WHAT IS ROBOTICS?

Robotics is the art, knowledge base, and the 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 to perform the necessary tasks. Robots may be used in manufacturing environments, in underwater and space exploration, for aiding the disabled, 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, 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 the recent history, one can see a close relationship between the state of industry, the revolution in numeric and computer control of machinery, space exploration, and the vivid imagination of creative people. Starting with Karel Čapek and his book, *Rossum's Universal Robots* [1], and continuing with movies like *Flash Gordon*, *Metropolis*, *Lost in Space*, *The Day The Earth Stood Still*, and *The Forbidden Planet* [2], we see that the stage was being set for a machine to be built to do human's job (and of course, R2D2, C3PO, and Robocop continued the trend). Čapek dreamt of a situation where a bioprocess could create human-like machines, devoid of emotions and souls, who were strong, obeyed 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 loss no one would care about. Eventually, the robots decided that they were actually superior to the humans and tried to take over the whole world. In this story, the word "rabota," or worker, was coined, and 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, for work on nuclear materials, multiple degree-of-freedom manipulators were being developed. A marriage between the NC capability of machine tools and the manipulators created a simple robot. The first robots were controlled by strips of paper with holes, which electric eyes could detect and which controlled the robot's movements. As industry improved, the strip of paper gave way to magnetic tapes, memory devices, and personal computers. The following is a summary of events that have marked changes in the direction of this industry:

- 1922 Czech author Karel Čapek wrote a story called *Rossum's Universal Robots* and introduced the word "Rabota" (meaning worker).
- 1946 George Devol developed the magnetic controller, a playback device. Eckert and Mauchley built the ENIAC computer at the University of Pennsylvania.

- 1952 The first NC machine was built at MIT.
- 1954 George Devol developed the first programmable robot.
- 1955 Denavit and Hartenberg developed homogeneous transformation matrices.
- 1961 U.S. patent 2,988,237 was issued to George Devol for “Programmed Article Transfer,” a basis for Unimate™ robots.
- 1962 Unimation was formed, first industrial robots appeared, and GM installed its first robot from Unimation.
- 1967 Unimate™ introduced the MarkII™ robot. The first robot was imported to Japan for paint-spraying applications.
- 1968 An intelligent robot called Shakey was built at 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. Westinghouse bought Unimation, which was later sold to Staubli of Switzerland.
- 1983 Robotics became a very popular subject, both in industry, as well as academia. Many programs in the nation started teaching courses in robotics.
- 1990 Cincinnati Milacron was acquired by ABB of Switzerland. Most small robot manufacturers went out of the market. Only a few large companies, which primarily produce industrial robots, remained.

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 without the need for life support, comfort, or concern about safety.
- Robots need no environmental comfort, such as lighting, air conditioning, ventilation, and noise protection.
- Robots work continuously without experiencing fatigue or boredom, 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 thousands of an inch. New wafer-handling robots have microinch accuracies.

- Robots and their accessories and sensors can have capabilities beyond that of humans.
- Robots can process multiple stimuli or tasks simultaneously. Humans can only process one active stimulus.
- Robots replace human workers creating economic problems, such as lost salaries, and social problems, such as dissatisfaction and resentment among workers.
- Robots lack 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 machines working with them [3]. This includes:
 - Inappropriate or wrong responses
 - A lack of decision-making power
 - A loss of power
 - Damage to the robot and other devices
 - Human injuries
- Robots, although superior in certain senses, have limited capabilities in
 - Degrees of freedom
 - Dexterity
 - Sensors
 - Vision systems
 - Real-time response
- Robots are costly, due to
 - Initial cost of equipment
 - Installation costs
 - 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 and consists of the links, the joints, and other structural elements of the robot. Without other elements, the manipulator alone is not a robot (Figure 1.2).

End effector This is the part that is connected to the last joint (hand) of a manipulator, which generally handles objects, makes connection to other machines, or performs the required tasks (Figure 1.2). 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 connecting specialty end effectors that are specifically designed for a purpose. This is the job of a company's engineers or

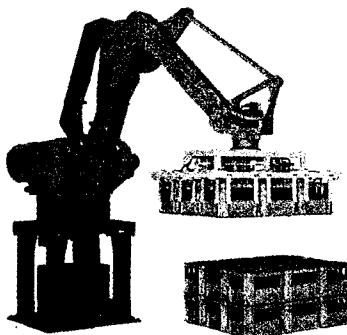


Figure 1.2 A Fanuc M-410iWW palletizing robotic manipulator with its end effector. (Reprinted by permission from Fanuc Robotics, North America, Inc.)

outside consultants to design and install the end effector on the robot and to make it work for the given situation. A welding torch, a paint spray gun, a glue-laying device, and a parts handler are but a few of the possibilities. In most cases, the action of the end effector is either controlled by the robot's controller, or the controller communicates with the end effector's controlling device (such as a PLC).

Actuators Actuators are the “muscles” of the manipulators. Common types of actuators are servomotors, stepper motors, pneumatic cylinders, and hydraulic cylinders. There are also other actuators that are more novel and are used in specific situations. This will be discussed in Chapter 6. Actuators are controlled by the controller.

Sensors Sensors are used to collect information about the internal state of the robot or to communicate with the outside environment. As in humans, the robot controller needs to know where each link of the robot is in order to know the robot's configuration. Even in absolute darkness, you still know where your arms and legs are! This is because feedback sensors in your central nervous system embedded in your muscle tendons send information to your brain. The brain uses this information to determine the length of your muscles, and thus, the state of your arms, legs, etc. The same is true for robots; sensors integrated into the robot send information about each joint or link to the controller, which determines the configuration of the robot. Robots are often equipped with external sensory devices such as a vision system, touch and tactile sensors, speech synthesizers, etc., which enable the robot to communicate with the outside world.

Controller The controller is rather similar to your cerebellum, and although it does not have the power of your brain, it still controls your motions. The controller receives its data from the computer, 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 will send 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 will then measure the change in the joint angle through the feedback sensor attached to the joint (a potentiometer, an encoder, etc.). When the joint reaches the desired value, the signal is stopped. In more sophisticated robots, the velocity 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, 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 a single purpose. It requires an operating system, programs, peripheral equipment such as monitors, and has many of the same limitations and capabilities of a PC processor.

Software There are perhaps three groups of software that are used in a robot. One is the operating system, which operates the computer. The second is the robotic software, which 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. The third group is the collection of routines and application programs that are developed in order to use the peripheral devices of the robots, such as vision routines, or to perform specific tasks.

It is important to note that in many systems, the controller and the processor are placed in the same unit. Although these two units are in the same box, and even if they are integrated into the same circuit, they have two separate functions.

1.8 ROBOT DEGREES OF FREEDOM

As you may remember from your engineering mechanics courses, in order to locate a point in space, one needs to specify three coordinates, such as the x -, y -, and z -coordinates along the three Cartesian axes. Three coordinates are necessary and sufficient to define the location of the point. Although the three coordinates may be expressed in terms of different coordinate systems, they are always necessary. However, it is not possible to have two or four coordinates, since two is inadequate to locate a point in space, and four is impossible in three dimensions. Similarly, if you consider a three-dimensional device with three degrees of freedom, within the workspace of the device, you should be able to place any point at any desired location. For example, a gantry (x,y,z) crane can place a ball at any location within its workspace as specified by the operator.

Similarly, to locate a rigid body (a three-dimensional object rather than a point) in space, one needs to specify the location of a selected point on it, and thus it requires three pieces of information to be located as desired. However, although the location of the object is specified, there are infinite possible ways to orientate the object about the selected point. To fully specify the object in space, in addition to the location of a selected point on it, one needs to specify the orientation of the object. This means that there is need for a total of six pieces of information to

fully specify the location and orientation of a rigid body. By the same token, there needs to be six degrees of freedom available to fully place the object in space and also orientate it as desired. If there are fewer than six degrees of freedom, the robot's capabilities are limited.

To demonstrate this, consider a robot with three degrees of freedom, 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 to pick up the part and to move it in space, parallel to the reference axes. The orientation always remains the same. Now consider another robot with five degrees of freedom, 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 -axes, but not z -axis.

A system with seven degrees of freedom does not have a unique solution. This means that if a robot has seven degrees of freedom, there are an infinite number of ways it can position a part and orientate it at the desired location. For the controller to know what to do, there must be some additional decision making routine that allows it to pick only one of the infinite ways. As an example, one 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 seven-degree-of-freedom 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 (Figure 1.3). The robot then has an additional degree of freedom, which, based on the preceding discussion, is impossible to control. The robot can be at a desired location and orientation from infinitely many distinct positions on the conveyor belt or the mobile platform. However, in this case, although there are too many degrees of freedom, generally, the additional degrees of freedom are not solved for. In other words, when a robot is mounted on a conveyor belt 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 degrees of freedom are still 6, and thus, unique. So long as the location of the base of the robot on the belt or the location of the mobile platform is known (or picked), there is no need to find it by solving a set of equations of robot motions, and, thus, the system can be solved.

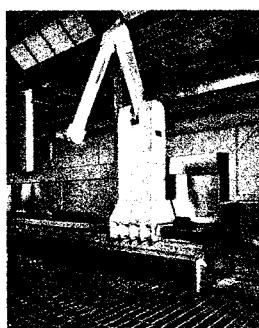


Figure 1.3 A Fanuc P-15 robot.
Reprinted with permission from Fanuc Robotics, North America, Inc.

Can you determine how many degrees of freedom the human arm has? This should exclude the hand (palm and the fingers), but should include the wrist. Before you go on, please try to see if you can determine it.

You will notice that the human arm has three joint clusters in it, the shoulder, the elbow and the wrist. The shoulder has three degrees of freedom, since the upper arm (humerus) can rotate in the sagittal plane (parallel to the mid-plane of the body), 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 three degrees of freedom. It can abduct and adduct, flex and extend, and since the radius bone can roll over the ulna bone, it can rotate longitudinally (pronate and supinate). Thus, the human arm has a total of seven degrees of freedom, even if the ranges of some movements are small. Since a seven-degree-of-freedom system does not have a unique solution, how do you think we can use our arms?

You must realize that in a robot system, the end effector is never considered as one of the degrees of freedom. 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 degrees of freedom.

There are cases where a joint may have the ability to move, but its movement is not fully controlled. For example, consider a linear joint actuated by a pneumatic cylinder, where the arm is fully extended or fully retracted, but no controlled position can be achieved between the two extremes. In this case, the convention is to assign only a 1/2-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 1/2 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 thus, has a limited degree of freedom.

There are many robots in industry that possess fewer than six degrees of freedom. In fact, robots with 3.5, 4, and 5 degrees of freedom are very common. So long as there is no need for the additional degrees of freedom, these robots perform very well. As an example, suppose that you desire to insert electronic components into a circuit board. The circuit board is always laid flat on a known work surface; thus, its height (z -value) relative to the base of the robot is known. Therefore, there is only need for two degrees of freedom along the x - and y -axes to specify any location on the board for insertion. Additionally, suppose that the components would be inserted in any direction on the board, but that the board is always flat. In that case, there will be need for one degree of freedom to rotate about the vertical axis (z) in order to orientate the component above the surface. Since there is also need for a 1/2-degree of freedom to fully extend the end effector to insert the part, or to fully retract it to lift the robot before moving, all that is needed is 3.5 degrees of freedom; two to move over the board, one to rotate the component, and 1/2 to insert or retract. Insertion robots are very common and are used extensively in electronic industry. Their advantage is that they are simple to program, are less expensive, and are 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 degrees of freedom 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, or spherical. Although spherical joints are common in many systems, since they possess multiple degrees of freedom, and thus, are difficult to control, spherical joints are not common in robotics, except in research. Most robots have either a linear (prismatic) joint or a rotary (revolute) joint.

Prismatic joints are linear; there is no rotation involved. They are either hydraulic or pneumatic cylinders, or they are linear electric actuators. These joints are used in gantry, cylindrical, or similar joint configurations.

Revolute joints are rotary, and although hydraulic and pneumatic rotary joints are common, most rotary joints are electrically driven, either by stepper motors or, more commonly, by servomotors.

1.10 ROBOT COORDINATES

Robot configurations generally follow the coordinate frames with which they are defined, as shown in Figure 1.4. 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's, R's, or S's. 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 are made of three linear joints that position the end effector, which are usually followed by additional revolute joints that orientate the end effector.

Cylindrical (R2P) Cylindrical coordinate robots have two prismatic joints and one revolute joint for positioning the part, plus revolute joints for orientating the part.

Spherical (2RP) 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 perhaps the most common configuration for industrial robots.

Selective Compliance Assembly Robot Arm (SCARA) SCARA robots have two revolute joints that are parallel and allow the robot to move in a horizontal plane, plus an additional prismatic joint that moves vertically (Figure 1.5). SCARA robots are very common in assembly operations. Their specific characteristic is that they are more compliant in the x - y -plane, but are very stiff along the

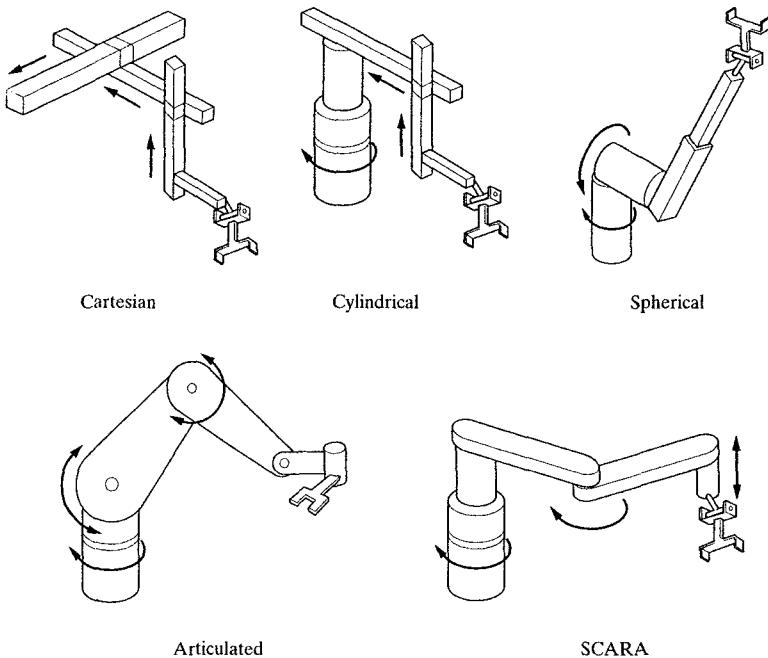


Figure 1.4 Some possible robot coordinate frames.

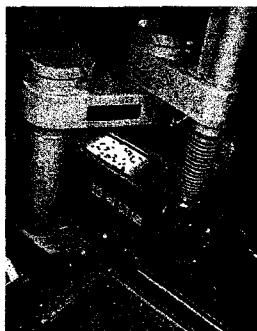


Figure 1.5 An Adept SCARA robot.
Reprinted with permission from Adept Technology, Inc.

z-axis, and thus have selective compliance. This is an important issue in assembly and will be discussed later.

1.11 ROBOT REFERENCE FRAMES

Robots may be moved relative to different coordinate frames. In each type of coordinate frame, the motions will be different. Usually, robot motions are accomplished in the following three coordinate frames (Figure 1.6):

World Reference Frame, which is a universal coordinate frame, as defined by x , y , z -axes. In this case, the joints of the robot move simultaneously so as to create motions along the three major axes. In this frame, for example, no matter where the arm is, a positive x -axis movement is always in the positive direction of the x -axis; this coordinate is used to define the motions of the robot relative to other objects, to define other parts and machines that the robot communicates with, and to define motion paths.

Joint Reference Frame, which is used to specify movements of each individual joint of the robot. Suppose that you want to move the hand of a robot to a particular position. You may decide to move one joint at a time in order to direct the hand to the desired location. In this case, each joint may be accessed individually, and, thus, only one joint moves at a time. Depending on the type of joint used (prismatic, revolute, or spherical), the motion of the robot hand will be different. For instance, if a revolute joint is moved, the hand will move around a circle defined by the joint axis.

Tool Reference Frame, which specifies movements of the robot's hand relative to a frame attached to the hand. The x' -, y' -, and z' -axes attached to the hand define the motions of the hand relative to this local frame. Unlike the universal World frame, the local Tool frame moves with the robot. Suppose that the hand is pointed as shown in Figure 1.6. Moving the hand relative to the positive x -axis of the local Tool frame will move the hand along the x' -axis of the Tool frame. If the arm were pointed elsewhere, the same motion along the local x' -axis of the Tool frame would be completely different from the first motion. The same $+x'$ -axis movement would be upward if the x' -axis were pointed upwards, and it would be downward if the x' -axis were pointed downward. As a result, the Tool reference frame is a moving frame that changes continuously as the robot moves, so 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 its sophistication. The following programming modes are very common:

Physical Setup In this mode, an operator sets up switches and hard stops that control the motion of the robot. This mode is usually used along with other devices, such as Programmable Logic Controllers (PLC).

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

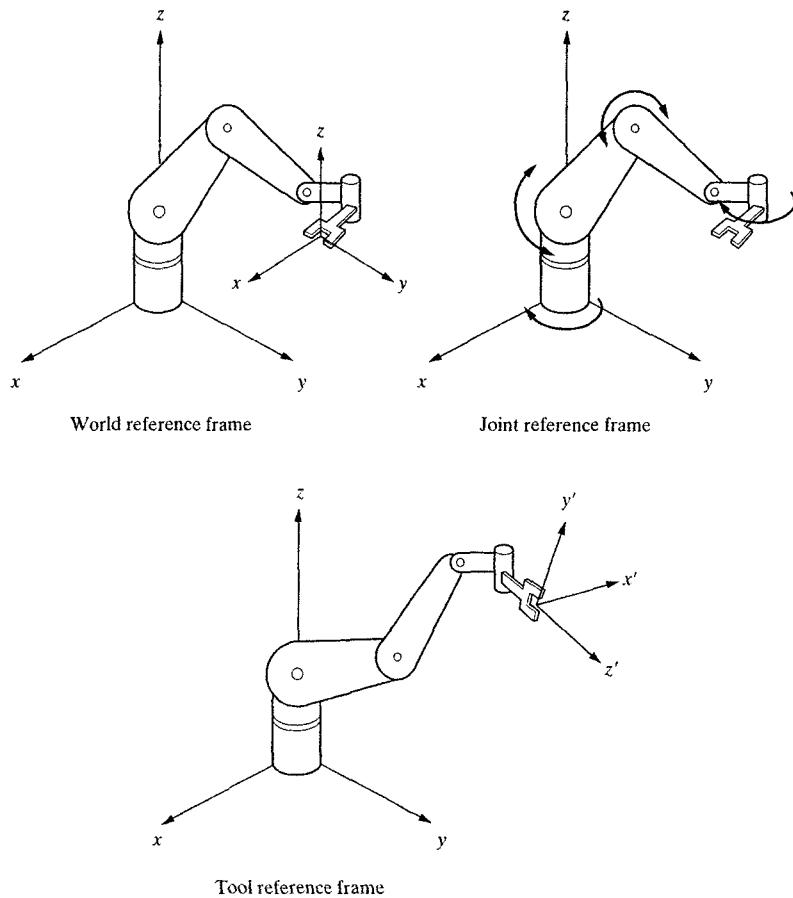


Figure 1.6 A robot's World, Joint, and Tool reference frames. Most robots may be programmed to move relative to either of these reference frames.

location is entered (taught) into the controller. During playback, the controller will move the joints to the same locations and orientations. This mode is usually point to point, where the motion between points is not specified or controlled. Only the points that are taught are guaranteed to reach.

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 directing the robot arm and moving it through its work-

space. Painting robots, for example, are programmed by skilled painters through this mode.

Software Mode In this mode of programming the robot, a program is written off-line or on-line 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 the knowledge of the operating system 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 Payload is the weight a robot can carry and still remain within its other specifications. For example, a robot's maximum load capacity may be much larger than its specified payload, but at the maximum level, it may become less accurate, may not follow its intended path accurately, or may have excessive deflections. The payload of robots compared with their own weight is usually very small. For example, Fanuc Robotics LR MateTM robot has a mechanical weight of 86 lbs and a payload of 6.6 lbs, and the M-16iTM robot has a mechanical weight of 594 lbs and a payload of 35 lbs.

Reach Reach is the maximum distance a robot can reach within its work envelope. As we will see later, many points within the work envelope of the robot may be reached with any desired orientation (called dexterous). However, for other points, close to the limit of robot's reach capability, orientation cannot be specified as desired (called nondexterous point). Reach is a function of the robot's joint lengths and its configuration.

Precision (validity) Precision is defined as how accurately a specified point can be reached. This is a function of the resolution of the actuators, as well as its feedback devices. Most industrial robots can have precision of 0.001 inch or better.

Repeatability (variability) Repeatability 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 same point every time, but will be within a certain radius from the desired point. The radius of a circle that is formed by this repeated motion is called repeatability. Repeatability is much more important than precision. If a robot is not precise, it will generally show a consistent error, which can be predicted and thus corrected through programming. As an example, suppose that a robot is consistently off 0.05 inch to the right. In that case, all desired points can be specified at 0.05 inch to the left, and thus the error can be eliminated. However, if the error is random, it cannot be predicted and thus cannot be eliminated.

Repeatability defines the extent of this random error. Repeatability is usually specified for a certain number of runs. More tests yield larger (bad for manufacturers) and 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 will be different from when the arm is tested in a horizontal configuration. Most industrial robots have repeatability in the 0.001 inch range.

1.14 ROBOT WORKSPACE

Depending on their configuration and the size of their links and wrist joints, robots can reach a collection of points called a workspace. The shape of the workspace for each robot is uniquely related to its characteristics. The workspace may be found mathematically by writing equations that define the robot's links and joints and including their limitations, such as ranges of motions for each joint [4]. Alternatively, the workspace may be found empirically, by moving each joint through its range of motions and combining all the space it can reach and subtracting what it cannot reach. Figure 1.7 shows the approximate workspace for some common configurations. When a robot is being considered for a particular application, its workspace must be studied to ensure that the robot will be able to reach the desired points. For accurate workspace determination, please refer to manufacturers' data sheets.

1.15 ROBOT LANGUAGES

There are perhaps as many robotic languages as there are robots. Each manufacturer designs its own robotic language, and thus, in order to use any particular

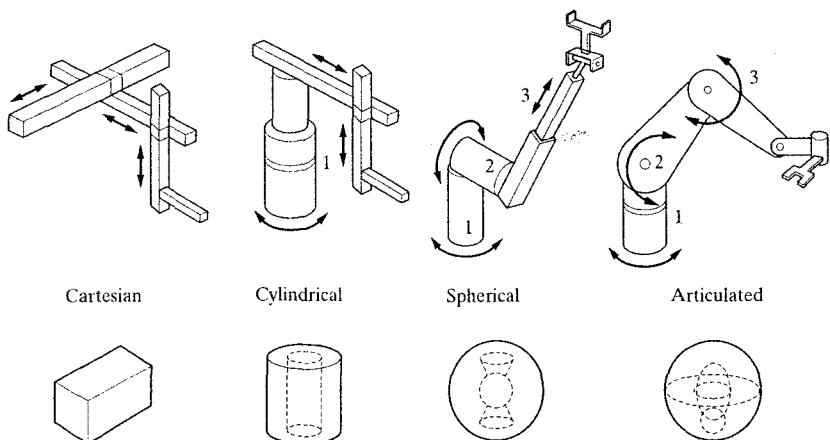


Figure 1.7 Typical workspaces for common robot configurations.

robot, its brand of programming language must be learned. Many robot languages are based on some other common language, such as Cobol, Basic, C, and Fortran. Other languages are unique and not directly related 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 [5,6,7]. High-level languages are either interpreter based or compiler based.

Interpreter-based languages execute one line of the program at a time, and each line has a line number. The interpreter interprets the line every time it is encountered (by converting the line to a machine language 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. 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. Thus, debugging programs 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 Unimation™ VAL® and IBM's AML® (A Manufacturing Language), are interpreter based [8,9].

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 during execution, 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 any error is present. As a result, debugging compiler-based programs is much more difficult. Certain languages, such as AL®, are more flexible. They allow the user to debug the program in interpreter mode, while the actual execution is in compiler mode.

The following is a general description of different levels of robotic languages [5]:

Microcomputer Machine Language Level In this level, the programs are written in machine language. This level of programming is the most basic and is very efficient, but difficult to understand and 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, which is easier to follow and understand.

Point-to-Point Level In this level (such as in Funky® and Cincinnati Milacron's T3®), 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; 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 (such as VAL by Unimation™). Most languages of this level are interpreter based.

Structured Programming Level Most languages of this level are compiler based, are powerful, and allow more sophisticated programming. However, they are also more difficult to learn.

Task-Oriented Level Currently, there are no actual languages of this level in existence. Autopass, proposed by IBM in the 1980s, never materialized. Autopass was supposed to be task oriented. This means that instead of programming a robot to perform a task by programming each and every step necessary to complete the task, the user was simply to mention the task, while the controller would create the necessary sequence. Imagine that a robot is to sort three boxes by size. In all existing languages, the programmer will have to tell the robot exactly what to do, which means that every step must be programmed. The robot must be told how to go to the largest box, how to pick up the box, where to place it, go to the next box, etc. In Autopass, the user would only indicate “sort,” while the robot controller would create this sequence automatically.

Example 1.1

The following is an example of a program written in VAL-II. This robotic language, released in 1979, is used with Unimation® and Puma® robots; it is interpreter based and allows for branching, sensory input, and output communication, straight-line movements, and many other features. For example, the user may define a distance “height” along the z-axis of the end effector that can be used with a command called APPRO (for approach) and DEPART in order to approach an object or depart from an object without collision. A command called MOVE will allow the robot to move from its present location to the next specified location. However, MOVES will do the same in a straight line. The difference is discussed in detail in Chapter 5. In the following listing, a number of different commands are described in order to show some of VAL-II’s capabilities:

1 .PROGRAM TEST	Declaration of the program name.
2 SPEED 30 ALWAYS	Sets the speed of the robot.
3 height=50	Specifies a distance for the liftoff and setdown points along the z-axis of the end effector.
4 MOVES p1	Moves the robot in 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 2 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 towards p6, but away from it a distance specified as "height," along the z-axis of the gripper (Tool frame). This is called a liftoff point.
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 z-axis of the gripper (Tool frame) a distance specified by "height."
17	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). The program is written for a 3P3R robot, with three prismatic linear positioning joints, three revolute orientation joints, and a gripper. Joints may be referred to by joint numbers <1, 2, 3, 4, 5, 6, 7>, where 1, 2, 3 indicate the prismatic joints; 4, 5, 6, indicates the revolute joints; and 7 indicates the gripper. The joints may also be referred to by index letters JX, JY, JZ, for motions along the x-, y-, z-axes, respectively, JR, JP, JY, for rotations about the Roll, Pitch, and Yaw axes (used for orientation), and JG, for the gripper.

There are two types of movements allowed in AML. MOVE commands are absolute. This means that the robot will move along the specified joint to the specified value. DMOVE commands are differential. This means that the joint will move the specified amount from wherever it is. Thus, MOVE (1, 10) means that the robot will move along the x-axis to 10 inches from the origin of the reference frame, whereas DMOVE (1, 10) means that the robot will move 10 inches along the x-axis from its current position. There is a large number of commands in AML, allowing the user to write sophisticated programs.

The following program will direct the robot to pick and place an object from one place to another. This is written to show you how a robotic program may be structured:

10	SUBR(PICK-PLACE);	Subroutine's name.
20	PT1: NEW <4., -24, 2, 0, 0, -13>;	Declaration of 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 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 a 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 z-axis.
100	DMOVE (7,-1.5);	Closes the gripper by 1.5 inches.
110	DMOVE (3, 1);	Moves up 1 inch along 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 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 to work in environments where humans cannot perform the tasks. Robots have already been used in many industries and for many purposes. They can often perform better than humans and at lower costs. For example, a welding robot can probably weld better than a human welder, because the robot can move more uniformly and more consistently. In addition, robots do not need protective goggles, protective clothing, ventilation, and many other necessities that their human counterparts do. As a result, robots can be more productive and better suited for the job, as long as the welding job is set up for the robot for automatic operations and nothing changes and as long as the welding job is not too complicated. Similarly, a robot exploring the ocean bottom would require far less attention than a human diver. Also, the robot can stay underwater for long periods and can go to very large depths and still survive the pressure; it also does not require oxygen.

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

Machine loading, where robots supply parts to or remove parts from other machines (Figure 1.8). In this type of work, the robot may not even perform any operation on the part, but is only a means of handling parts within a set of operations.

Pick and place operations, where the robot picks up parts and places them elsewhere (Figure 1.9). This may include palletizing, placing 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 part from the oven, or other similar routines.

Welding, where 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 the robots' consistent movements, the welds are



Figure 1.8 A Staubli robot loading and unloading components into and from a machining center. Reprinted with permission from Staubli Robotics.



Figure 1.9 Staubli robots placing dishwasher tubs into welding stations.
Reprinted with permission from Staubli Robotics.



Figure 1.10 An AM120 Fanuc robot.
Reprinted with permission from Fanuc Robotics, North America, Inc.

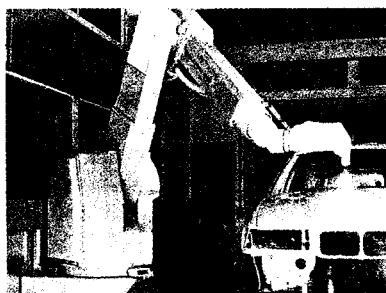


Figure 1.11 A P200 Fanuc robot painting automobile bodies.
Reprinted with permission from Fanuc Robotics, North America, Inc.

very uniform and accurate. Welding robots are usually large and powerful (Figure 1.10).

Painting is another very common application of robots, especially in the automobile industry. Since maintaining a ventilated, but clean, room suitable for humans is difficult and compared with those performed by humans, robotic operations are more consistent, painting robots are very well suited for their job (Figure 1.11).

Inspection of parts, circuits boards, and other similar products is also a very common application for robots. In general, some other device is integrated into the system for inspection. This may be a vision system, an X-ray device, an ultrasonic detector, or other similar devices (Figure 1.12). In one application, a robot equipped

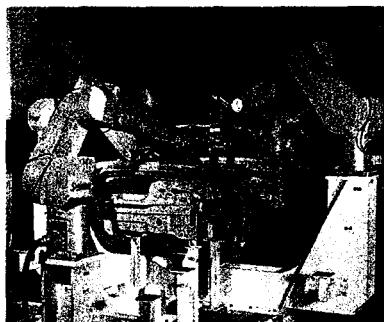


Figure 1.12 Staubli RX FRAMS (Flexible Robotic Absolute Measuring System) robots in a BMW manufacturing facility. Reprinted with permission from Staubli Robotics.

with an ultrasound crack detector was given the computer-aided design (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 was used to search for and find the location of each rivet, detect and mark the rivets with fatigue 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 such applications, including part identification, the characteristics (such as the circuit diagram of a circuit board, the nameplate of a part, etc.) of the part are stored in the system in a data library. The system uses this information to match the part with the stored data. The part is either accepted or rejected, based on the result of the inspection.

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

Assembly operations are among the most difficult for the robot to do. Usually, assembling components into a product involves many operations. For example, the parts must be located and identified, carried in a particular order with many obstacles around the setup, fitted together, and then assembled. Many of the fitting and assembling tasks are complicated as well, and may require pushing, turning, bending, wiggling, pressing, and snapping the tabs to connect the parts. 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.

Manufacturing by robots may include many different operations, such as material removal (Figure 1.13), drilling, deburring, laying glue, cutting, etc. It also includes insertion of parts, such as electronic components into circuit boards, installation of boards into electronic devices such as VCR's, and other similar operations. Insertion robots are also very common and are extensively used in electronic industry.

Surveillance by robots has been tried, but was not too successful. However, the use of vision systems for surveillance has been very extensive, both in security industry and in traffic control. For example, in one part of the highway system in

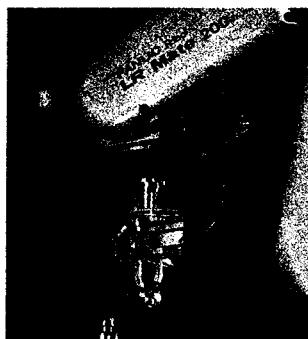


Figure 1.13 A Fanuc LR Mate 200i robot is used in a material removal operation on a piece of jewelry. Reprinted with permission from Fanuc Robotics, North America, Inc.

Southern California, one lane of traffic has been leased out to private industry, which maintains the road and provides services, but also charges users. Surveillance cameras are used to detect the license plates of the cars that use the road, which are subsequently charged a toll for road use.

Medical applications are also becoming increasingly common. For example, the Robodoc was designed to assist a surgeon in total-joint-replacement operations. Since many of the functions that are performed during this procedure, such as cutting of the head of the bone, drilling a hole in the bone's body, reaming the hole for precise dimension, and installation of the manufactured implant joint, can be performed by a robot with better precision than by a human, the mechanical parts of the operation are assigned to the robot. In addition, the orientation and the shape of the bone can be determined by CAT-scan and downloaded to the robot controller, where it is used to direct the motions of the robot for best fit with the implant. Similarly, many other robots have been used to assist surgeons during microsurgery, including operation on heart valves in Paris and Liepzig [10]. Another robot called da Vinci Surgical Robot, which is approved by U.S. Food and Drug Administration (FDA), was used to perform abdominal surgery [11].

Assisting disabled individuals has also been tried with interesting results. There is much that can be done to help the disabled in their daily lives. In one study, a small table-top robot was programmed to communicate with a disabled person and to perform simple tasks such as placing a food plate into the microwave oven, removing the plate from the oven, and placing the plate in front of the disabled person to eat [12]. Many other tasks were also programmed for the robot to perform.

Hazardous environments are well suited for robotics use. Because of their inherent danger in these environments, humans must be well protected against the dangers. However, robots can access, traverse, maintain, and explore these areas without the same level of concern. Servicing a radioactive environment, for instance, can be done much easier with a robot than with a human. In 1993, an eight-legged robot called Dante was to reach the lava lake of constantly erupting volcano of Mount Erebus in Antarctica and study its gases [13].



Figure 1.14 The Arm, a six-degree-of-freedom bilateral force-feedback manipulator, used primarily on manned submersibles and remotely operated vehicles. Reprinted with permission from Western Space and Marine, Inc.

Underwater, space, and remote locations can also be serviced or explored by robots. Although no human has yet been sent to Mars, there have been a number of rovers that have already landed and explored it [14]. The same is true for other space and underwater applications [15,16,17]. Until recently, for example, very few sunken ships were explored in deep oceans, because no one could access those depths. Many crashed airplanes, as well as sunken ships and submarines, are nowadays recovered quickly by underwater robots.

In an attempt to clean the smudge from inside of a steam generator blowdown pipe, a teleoperated robot called Cecil was designed to crawl down the pipe and wash away the smudge with a stream of water at 5,000 psi [18]. Figure 1.14 shows The Arm, a six-degree-of-freedom bilateral force-feedback manipulator, used primarily on manned submersibles and remotely operated vehicles. The Arm is controlled via a remote master that also “feels” everything that the slave arm “feels.” The system can also perform preprogrammed motions through a teach-and-repeat system.

In another application, a telerobot was used for microsurgery [19]. In this case, the location of the telerobot is of secondary concern. The primary intention is to have the telerobot repeat the surgeon’s hand movements at smaller scale for reduced tremor during microsurgery.

1.17 OTHER ROBOTS AND APPLICATIONS

With the same interest that helped scientists and engineers design human-like robots, other robots have been designed to imitate insects and other animals. Examples include six- and eight-legged robots [20,21], wormlike robots [22], snake-like robots [23,24], robots that swim like a fish [25], a robot that behaves like a dog, a Lobster-like robot [26], and unidentified life forms [27]. Some of these robots, such as Odetics, Inc. Odex robot [28], are very large and powerful; others are very small and lightweight. Most of these robots are developed for research purposes.

However, others are designed to be used in military operations [28], in medical operations, or for entertainment. In one case, a small robotic mine-sweeper was developed to search for mines and to explode them. The rationale is that it is far better to destroy a low-cost robot in exploding a mine than it is to lose a life or have casualties [29].

Animatronics refers to the design and development of systems that are used in animated robotic figures and machines that look and behave like humans and other animals. Examples include animatronic lips [30], eyes [31], and hands. As more sophisticated animatronic components become available, the action figures they replace become increasingly real.

Another area that is somewhat related to robotics and its applications is Micro-Electro-Mechanical-Systems (MEMS). These are microlevel devices that are designed to perform functions within a system, which may include medical, mechanical, electrical, and physical tasks. For example, a microlevel 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, or a MEMS actuator may be used to deploy automobile airbags in a collision [32,33].

1.18 SOCIAL ISSUES

One has to always consider the social consequences that may result from using robots. Although there are many applications for which robots are used because there are no workers who can do the same job, there are many other applications for which a robot replaces a human worker. The worker who is replaced by a robot will lose his or her income. If the trend continues without consideration, it is conceivable that there may be a situation where most products are made by robots, without the need for any human workers. The result will be increasingly fewer workers with jobs, who lack the money to buy the products the robots make. More importantly, the issue to consider is the social and economic problems that arise as increasingly more workers become unemployed. One of the important points of negotiations between the automobile manufacturers and the United Auto Workers (UAW) is how many human jobs, and at what rate, may be replaced by robots.

Although no solution is presented in this writing, many references are available for further study of the problem [34,35]. However, as an engineer who strives to make better products at lower costs, and who may consider the use of robots to replace a human worker, one must always remember the consequences of this choice. Our academic and professional interest in robotics must always be intertwined with the social and economic considerations of robotics.

1.19 SUMMARY

Many people interested in robotics have some knowledge about robots, and, in many cases, they have had some interaction with robots. However, it is necessary that certain ideas be understood by everyone. In this chapter, we discussed some

fundamental ideas about robotics that enable us to better understand situations in which robots might be used. Robots can be used for many purposes, including industrial applications, entertainment, and other specific and unique applications such as in space, underwater, and in hazardous environments. Obviously, as time goes by, robots will be used for more applications. The remainder of this book will discuss the kinematic and kinetics of robots, their components (such as actuators, sensors, and vision systems), and their applications.

REFERENCES

1. Čapek, Karel, *Rossum's Universal Robots*, translated by Paul Selver, Doubleday, New York, 1923.
2. Valenti, Michael, "A Robot Is Born," *Mechanical Engineering*, June 1996, pp. 50–57.
3. Bonney, M. C., and Y. F. Yong, editors, *Robot Safety*, IFS Publications, Ltd., UK, 1985.
4. Wiitala, Jared M., B. J. Rister, and J. P. Schmiedler, "A More Flexible Robotic Wrist," *Mechanical Engineering*, July 1997, pp. 78–80.
5. Bonner, Susan, K. G. Shin, "A Comprehensive Study of Robot Languages," *IEEE Computer*, December 1982, pp. 82–96.
6. Kusiak, Andrew, "Programming, Off-Line Languages," *International Encyclopedia of Robotics: Applications and Automation*, Richard C. Dorf, Editor, John Wiley & Sons, New York, 1988, pp. 1235–1250.
7. Gruber, William, B. I. Soroka, "Programming, High Level Languages," *International Encyclopedia of Robotics: Applications and Automation*, Richard C. Dorf, Editor, John Wiley & Sons, New York, 1988, pp. 1203–1234.
8. Unimation, Inc., *VAL-II Programming Manual*, Version 4, Pittsburgh, 1988.
9. IBM Corp., *AML Concepts and User's Guide*, 1983.
10. Salisbury, Kenneth, Jr., "The Heart of Microsurgery," *Mechanical Engineering*, December 1998, pp. 46–51.
11. "da Vinci Surgical Robot," cited in the *Today Show*, NBC News, July 13, 2000.
12. "Stanford Rehabilitation Center," Stanford University, California.
13. Leary, Warren, "Robot Named Dante To Explore Inferno of Antarctic Volcano," *The New York Times*, December 8, 1992, p. B7.
14. <http://www.jpl.nasa.gov/pictures/>.
15. Wernli, Robert L., "Robotics Undersea," *Mechanical Engineering*, August 1982, pp. 24–31.
16. Asker, James, "Canada Gives Station Partners A Hand – And An Arm," *Aviation Week & Space Technology*, December 1997, pp. 71–73.
17. Puttre, Michael, "Space-Age Robots Come Down to Earth," *Mechanical Engineering*, January 1995, pp. 88–89.
18. Trovato, Stephen A., "Robot Hunts Sludge and Hoses It Away," *Mechanical Engineering*, May 1988, pp. 66–69.
19. "Telerobot Control for Microsurgery," *NASA Tech Briefs*, October 1997, p. 46.
20. Yeaple, Judith A., "Robot Insects," *Popular Science*, March 1991, pp. 52–55 and 86.
21. Freedman, David, "Invasion of the Insect Robots," *Discover*, March 1991, pp. 42–50.
22. Thakoor, Sarita, B. Kennedy, A. Thakoor, "Insectile and Vemiform Exploratory Robots," *NASA Tech Briefs*, November 1999, pp. 61–63.

23. "Snakelike Robots Would Maneuver in Tight Spaces," *NASA Tech Briefs*, August 1998, pp. 36–37.
24. Terry, Bryan, "The Robo-Snake," *Senior Project Report*, Cal Poly, San Luis Obispo, CA, June 2000.
25. O'Conner, Leo, "Robotic Fish Gotta Swim, Too," *Mechanical Engineering*, January 1995, p. 122.
26. Chalmers, Peggy, "Lobster Special," *Mechanical Engineering*, September 2000, pp. 82–84.
27. Lipson, Hod, J. B. Pollack, "The Golem Project: Automatic Design and Manufacture of Robotic Lifeforms," <http://golem03.cs.i.brandies.edu/index.html>.
28. Corrado, Joseph K., "Military Robots," *Design News*, October 83, pp. 45–66.
29. "Low-Cost Minesweeping," *Mechanical Engineering*, April 1996, p. 66.
30. #2 Acorn Centre, 30–34 Gorst Road, Park Royal, London NW10 6LE, England. e-mail: info@gems-figures.co.uk.
31. 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.
32. O'Conner, Leo, "MEMS: Microelectromechanical Systems," *Mechanical Engineering*, February 1992, pp. 40–47.
33. O'Conner, Leo, H. Hutchinson, "Skyscrapers in a Microworld," *Mechanical Engineering*, March 2000, pp. 64–67.
34. Coates, V. T., "The Potential Impacts of Robotics," Paper number 83-WA/TS-9, *American Society of Mechanical Engineers*, 1983.
35. Albus, James, "Brains, Behavior, and Robotics," *Byte Books*, McGraw-Hill, 1981.

PROBLEMS

1. Draw the approximate work space for the robot shown in Figure P.1.1. Assume that the dimensions of the base and other parts of the structure of the robot are as shown.
2. Draw the approximate work space for the robot shown in Figure P.1.2. Assume that the dimensions of the base and other parts of the structure of the robot are as shown.

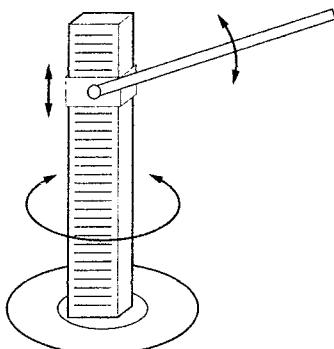


Figure P.1.1

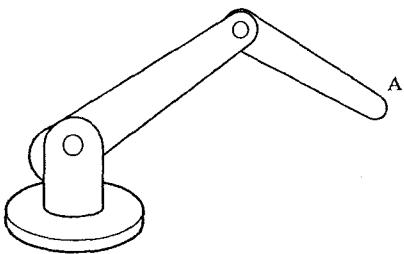


Figure P.1.2

3. Draw the approximate workspace for the robot shown in Figure P.1.3. Assume that the dimensions of the base and other parts of the structure of the robot are as shown.

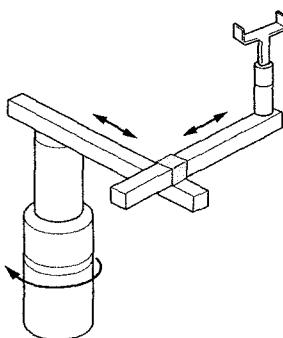


Figure P.1.3

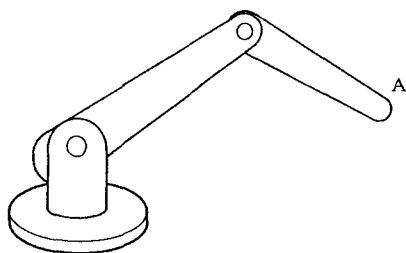


Figure P.1.2

3. Draw the approximate workspace for the robot shown in Figure P.1.3. Assume that the dimensions of the base and other parts of the structure of the robot are as shown.

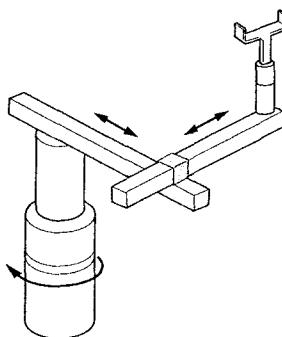


Figure P.1.3

2

Robot Kinematics: Position Analysis

2.1 INTRODUCTION

In this chapter, we will study forward and inverse kinematics of robots. The forward kinematics will enable us to determine where the robot's end (hand) will be if all joint variables are known. Inverse kinematics will enable us to calculate what each joint variable must be if we desire that the hand be located at a particular point and have a particular orientation. Using matrices, we will first establish a way of describing objects, locations, orientations, and movements. Then we will study the forward and inverse kinematics of different configurations of robots, such as Cartesian, cylindrical, and spherical coordinates. Finally, we will use the Denavit–Hartenberg representation to derive forward and inverse kinematic equations of all possible configurations of robots.

It is important to realize that in reality, 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 is. For a short end effector, the end will be at a different location than for a long end effector. In this chapter, we will 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.

2.2 ROBOTS AS MECHANISMS

Manipulator-type robots have multiple degrees of freedom (DOF), are three-dimensional, are open loop, and are chain mechanisms.

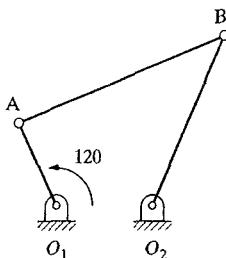


Figure 2.1 A one-degree-of-freedom closed-loop four-bar mechanism.

In a one-degree-of-freedom system, when the variable is set to a particular value, the mechanism is totally set, and all its other variables are known. For example, in the four-bar mechanism of Figure 2.1, when the crank is set to 120°, the angles of the coupler link and the rocker arm are also known. However, in a multiple-degree-of-freedom mechanism, all input variables must be individually set in order to know the remaining parameters. Robots are such machines, where each joint variable must be known in order to know where the hand of the robot is.

Robots are three-dimensional machines if they are to move in space. Although it is possible to have a two-dimensional multiple-degree-of-freedom robot, they are not common.

Robots are open-loop mechanisms. Unlike mechanisms that are closed loop (e.g., four-bar mechanisms), even if all joint variables are set to particular values, there is no guarantee that the hand will be at the given location. This is because if there is any deflection in any joint or link, it will change the location of all subsequent links without feedback. For example, in the four-bar mechanism of Figure 2.2, if the link AB deflects, it will affect link O_2B , whereas in an open-loop system such as the robot, the deflections will move all succeeding members without any feedback. As a result, in open-loop systems, either all joint and link parameters must continuously be measured or the end of the system must be monitored for the kinematic position of the machine to be known. This difference can be expressed by comparing the vector equations describing the relationship between different links of the two mechanisms as follows:

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

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

As you can see, if there is a deflection in link AB , link O_2B will move accordingly. However, the two sides of Equation (2.1) have changed 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, 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 is the standard practice, the robot's links and joints are made excessively strong to eliminate all deflections. The last method renders the robot very heavy,

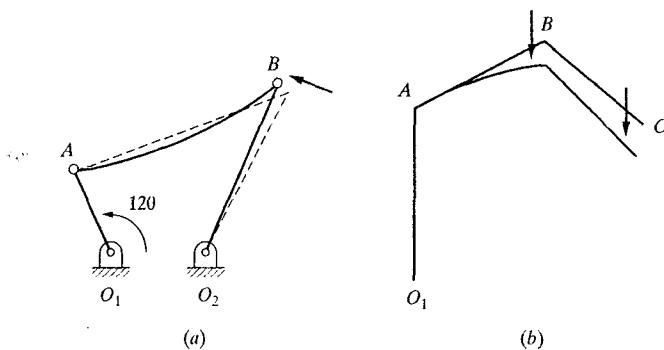


Figure 2.2 (a) Closed-loop versus (b) open-loop mechanisms.

massive, and slow, and its specified payload is very low compared with what it actually can carry.

2.3 MATRIX REPRESENTATION

Matrices can be used to represent points, vectors, frames, translations, rotations, and transformations, as well as objects and other kinematic elements in a frame. We will use this representation throughout this book to derive equations of motion for robots.

2.3.1 Representation of a Point in Space

A point P in space (Figure 2.3) can be represented by its three coordinates relative to a reference frame:

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

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.

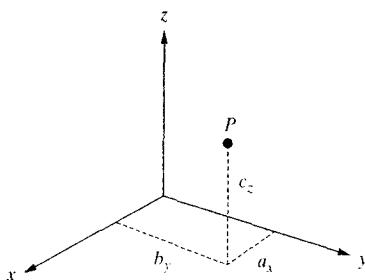


Figure 2.3 Representation of a point in space.

2.3.2 Representation of a Vector in Space

A vector can be represented by three coordinates of its tail and of its head. If the vector starts at a point A and ends at point B , then it can be represented by $\bar{P}_{AB} = (B_x - A_x)\hat{i} + (B_y - A_y)\hat{j} + (B_z - A_z)\hat{k}$. Specifically, if the vector starts at the origin (Figure 2.4), then:

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

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 previous section 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 a matrix form, as in Equation (2.5). This format will be used throughout this book to represent all kinematic elements:

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

This representation can be slightly modified to also include a scale factor w such that if x , y , and z are divided by w , they will yield a_x , b_y , and c_z . Thus, the vector can be written as

$$\bar{P} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}, \quad \text{where } a_x = \frac{x}{w}, b_y = \frac{y}{w}, \text{ etc.} \quad (2.6)$$

Variable w may be any number, and as it changes, it can change the overall size of the vector. This is similar to zooming a picture in computer graphics. As the value of w changes, the size of the vector changes accordingly. If w is greater than unity, all vector components enlarge; if w is less than unity, all vector components become smaller. This is also used in computer graphics for changing the size of pictures and drawings.

If w is unity, the size of the components remain unchanged. However, if $w = 0$, then a_x , b_y , and c_z will be infinity. In this case, x , y , and z (as well as a_x , b_y , and c_z) will

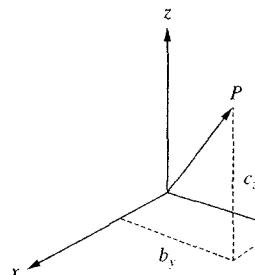


Figure 2.4 Representation of a vector in space.

represent a vector whose length is infinite, but nonetheless, is in the direction represented by the vector. This means that a **directional vector** can be represented by a scale factor of $w = 0$, where the length is not of importance, but the direction is represented by the three components of the vector. This will be used throughout this book to represent directional vectors.

Example 2.1

A vector is described as $\bar{P} = 3\hat{i} + 5\hat{j} + 2\hat{k}$. 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 0 for direction, as

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

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

$$\lambda = \sqrt{p_x^2 + p_y^2 + p_z^2} = 6.16, \quad \text{where } p_x = \frac{3}{6.16} = 0.487, p_y = \frac{5}{6.16}, \text{ etc.,}$$

$$\text{and} \quad \bar{P}_{\text{unit}} = \begin{bmatrix} 0.487 \\ 0.811 \\ 0.324 \\ 0 \end{bmatrix}.$$

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

A frame centered at the origin of a reference frame is represented by three vectors, usually mutually perpendicular to each other, called unit vectors $\bar{n}, \bar{o}, \bar{a}$, for *normal*, *orientation*, and *approach* vectors (Figure 2.5). Each unit vector is represented by its

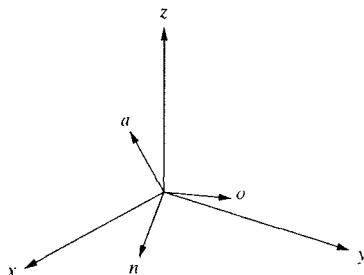


Figure 2.5 Representation of a frame at the origin of the reference frame.

three components in the reference frame as in Section 2.3.2. Thus, a frame F can be represented by three vectors in a matrix form as:

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

2.3.4 Representation of a Frame in a Fixed Reference Frame

If a frame is not at the origin (or, in fact, even if it is at the origin) then the location of the origin of the frame relative to the reference frame must also be expressed. In order to do this, a vector will be drawn between the origin of the frame and the origin of the reference frame describing the location of the frame (Figure 2.6). This vector is expressed through its components relative to the reference frame. Thus, the frame can be expressed by three vectors describing its directional unit vectors, as well as a fourth vector describing its location as follows:

$$F = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.8)$$

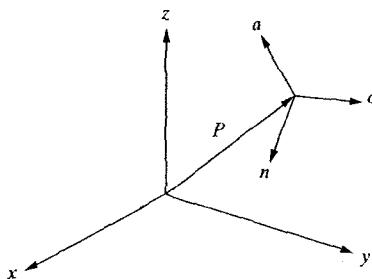


Figure 2.6 Representation of a frame in a frame.

As shown in Equation (2.8), the first three vectors are directional vectors with $w = 0$, representing the directions of the three unit vectors of the frame $\bar{n}, \bar{o}, \bar{a}$, 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 to us, and thus we use a scale factor of one. A frame may also be represented with a 3×4 matrix without the scale factors, but it is not commonly done this way.

Example 2.2

The frame F shown in Figure 2.7 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:

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

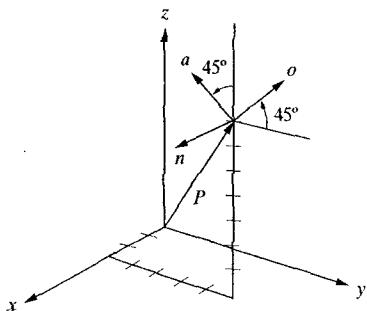


Figure 2.7 An example of representation of a frame in space.

2.3.5 Representation of a Rigid Body

An object can be represented in space by attaching a frame to it and representing the frame in space. Since the object is permanently attached to this frame, its position and orientation relative to this frame is always known. 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 2.8). As before, a frame in space can be represented by a matrix, where the origin of the frame, as well as the three vectors representing its orientation relative to the reference frame, are expressed. Thus,

$$F_{\text{object}} = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.9)$$

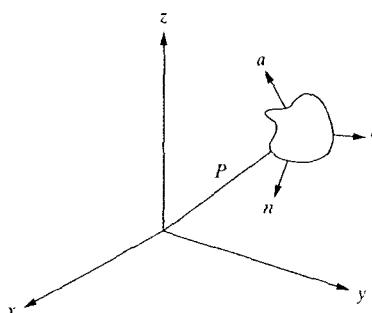


Figure 2.8 Representation of an object in space.

As was discussed in Chapter 1, a point in space has only three degrees of freedom; it can only move along the three reference axes. However, a rigid body in space has 6 degrees of freedom, meaning that not only it can move along three axes of x , y , and z , but can also rotate about these three axes. Thus, all that is needed to completely define an object in space is 6 pieces of information describing the location of the origin of the object in the reference frame relative to the three reference axes, as well as its orientation about the three axes. However, as can be seen in Equation (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 preceding to 6. Thus, we need 6 constraint equations to reduce the amount of information from 12 to 6 pieces. The constraints come from the known characteristics of the frame, which we have not used yet:

- the three unit vectors $\bar{n}, \bar{o}, \bar{a}$ are mutually perpendicular, and
- each unit vector's length must be equal to unity.

These constraints translate into the following six constraint equations:

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

As a result, the values representing a frame in a matrix must be such that the foregoing equations are true. Otherwise, the frame will not be correct. Alternatively, the first three equations in Equation (2.10) can be replaced by a cross product of the three vectors as follows:

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

Example 2.3

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, the values 5,3,2 representing the position of the origin of the frame do not affect the constraint equations. You also notice that only three values for directional vectors are given. This is all that is needed. Using Equation (2.10), we get

$$n_x o_x + n_y o_y + n_z o_z = 0, \quad \text{or} \quad n_x (0) + 0.707 (o_y) + n_z (o_z) = 0,$$

$$n_x a_x + n_y a_y + n_z a_z = 0, \quad \text{or} \quad n_x (a_x) + 0.707 (a_y) + n_z (0) = 0,$$

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

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 $a_y = -0.707$. Please 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 will be:

$$F = \begin{bmatrix} 0.707 & 0 & 0.707 & 5 \\ 0.707 & 0 & -0.707 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{or} \quad F = \begin{bmatrix} -0.707 & 0 & -0.707 & 5 \\ 0.707 & 0 & -0.707 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

As can be seen, both matrices satisfy all the requirements set by the constraint equations. It is important to realize that the values represented by the three direction vectors are not arbitrary, but are bound by these equations. Thus, you may not arbitrarily use any desired values in the matrix.

The same problem may be solved by taking the cross products of \bar{n} and \bar{o} and setting it equal to \bar{a} as $\bar{n} \times \bar{o} = \bar{a}$, or

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

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

Substituting the values into this equation yields:

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

Solving the three simultaneous equations gives:

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

which replace the three equations for the dot products. Together with the three unit vector length constraint equations, there are six equations, which ultimately result in the same values for the unknown parameters. Please verify that you get the same results.

2.4 HOMOGENEOUS TRANSFORMATION MATRICES

For a variety of reasons, it is desirable to keep transformation matrices in square form, either 3×3 or 4×4 . First, as we will 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, one may multiply A by B or B by A , both resulting in the same $(m \times m)$ 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 desire square matrices.

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

$$F = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.12)$$

2.5 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 can represent this motion in a form similar to a frame representation. This is because a transformation itself is a change in the state of a frame (representing the change in its location and orientation), and thus it can be represented as 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 or rotations.

To see how these can be represented, we will study each one separately.

2.5.1 Representation of a Pure Translation

If a frame (which may also be representing 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 in the same direction and thus do not change. All that changes is the location of the origin of the frame relative to the reference frame, as shown in Figure 2.9. The new location of the frame relative to the fixed reference frame can be found by adding the vector representing the translation to the vector representing the original location of the origin of the frame. In matrix form, the new frame representation may be found by premultiplying the frame with a matrix representing the transformation. Since the directional vectors do not change in a pure translation, the transformation T will simply be:

$$T = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.13)$$

where d_x , d_y , and d_z are the three components of a pure translation vector \bar{d} relative to the x -, y -, and z -axes of the reference frame. As you can see, the first three columns represent no rotational movement (equivalent to unity), while the last column represents the translation. The new location of the frame will be

$$F_{\text{new}} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & P_x + d_x \\ n_y & o_y & a_y & P_y + d_y \\ n_z & o_z & a_z & P_z + d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.14)$$

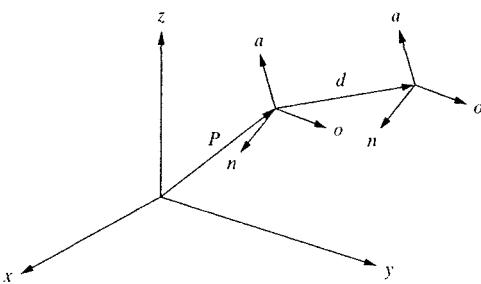


Figure 2.9 Representation of a pure translation in space.

This equation is also symbolically written as

$$F_{\text{new}} = \text{Trans}(d_x, d_y, d_z) \times F_{\text{old}}. \quad (2.15)$$

First, as you see, by premultiplying the transformation matrix with the frame matrix, the new location can be found. This, in one form or another, is true for all transformations, as we will see later. Second, you notice that the directional vectors remain the same after a pure translation, but that as vector addition of d and P would result, the new location of the frame is $d + P$. Third, you also notice how homogeneous transformation matrices facilitate the multiplication of matrices, resulting in the same dimensions as before.

Example 2.4

A frame F has been moved nine units along the x -axis and five units along the z -axis of the reference frame. Find the new location of the frame:

$$F = \begin{bmatrix} 0.527 & -0.574 & 0.628 & 5 \\ 0.369 & 0.819 & 0.439 & 3 \\ -0.766 & 0 & 0.643 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Solution Using Equation (2.14) or Equation (2.15), we get

$$\begin{aligned} F_{\text{new}} &= \text{Trans}(d_x, d_y, d_z) \times F_{\text{old}} = \text{Trans}(9, 0, 5) \times F_{\text{old}} \\ \text{and } F &= \begin{bmatrix} 1 & 0 & 0 & 9 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.527 & -0.574 & 0.628 & 5 \\ 0.369 & 0.819 & 0.439 & 3 \\ -0.766 & 0 & 0.643 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0.527 & -0.574 & 0.628 & 14 \\ 0.369 & 0.819 & 0.439 & 3 \\ -0.766 & 0 & 0.643 & 13 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

2.5.2 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 parallel to it. We will later expand the results to other rotations, as well as combination of rotations.

Let's assume that a frame $(\bar{n}, \bar{o}, \bar{a})$, located at the origin of the reference frame $(\bar{x}, \bar{y}, \bar{z})$, will rotate through an angle of θ about the x -axis of the reference frame. Let's also assume that attached to the rotating frame $(\bar{n}, \bar{o}, \bar{a})$ is a point P , with coordinates P_x , P_y , and P_z relative to the reference frame and P_n , 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_n , P_o , and P_a coordinates of the point remain

the same in the rotating frame $(\bar{x}, \bar{o}, \bar{a})$, but P_x , P_y , and P_z will be different in the $(\bar{x}, \bar{y}, \bar{z})$ frame (Figure 2.10). We desire 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 2-D as if we were standing on the x -axis. The coordinates of point P are shown before and after rotation in Figure 2.11. The coordinates of point P relative to the reference frame are P_x , P_y ,

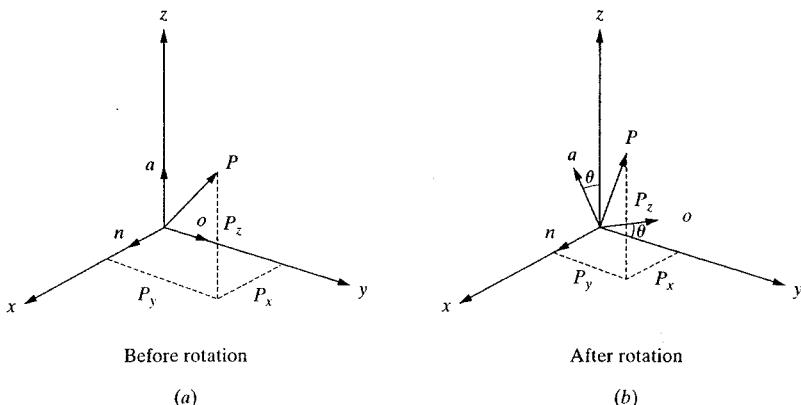


Figure 2.10 Coordinates of a point in a rotating frame before and after rotation.

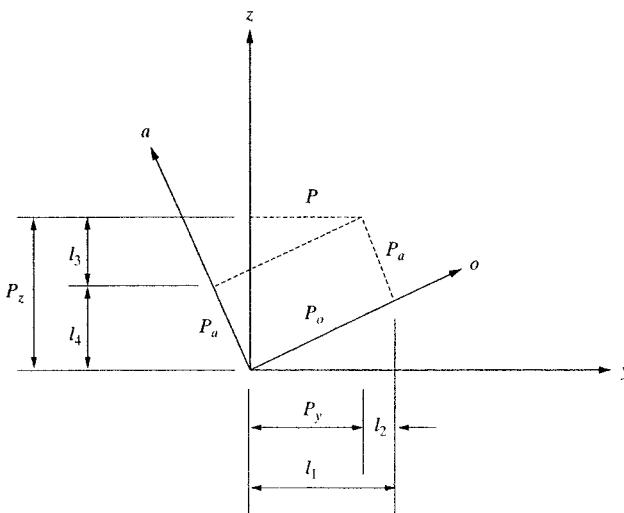


Figure 2.11 Coordinates of a point relative to the reference frame and rotating frame as viewed from the x -axis.

and P_z , while its coordinates relative to the rotating frame (to which the point is attached) remain as P_n , P_o , and P_a .

From Figure 2.11, you will see 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. Please verify that

$$\begin{aligned} P_x &= P_n, \\ P_y &= l_1 - l_2 = P_o \cos \theta - P_a \sin \theta, \\ P_z &= l_3 + l_4 = P_o \sin \theta + P_a \cos \theta, \end{aligned} \quad (2.16)$$

which is in matrix form

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} P_n \\ P_o \\ P_a \end{bmatrix}. \quad (2.17)$$

This means that the coordinates of the point (or vector) 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

$$P_{xyz} = \text{Rot}(x, \theta) \times P_{noa}. \quad (2.18)$$

Please also notice that the first column of the rotation matrix in Equation (2.16), which expresses the location relative to the x -axis, has 1,0,0 values, indicating that the coordinate along the x -axis has not changed.

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

$$\text{Rot}(x, \theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\theta & -S\theta \\ 0 & S\theta & C\theta \end{bmatrix}. \quad (2.19)$$

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

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

Equation (2.18) can also be written in a conventional form, which assists in easily following the relationship between different frames. Denoting the transformation as ${}^U T_R$ (and reading it as the transformation of frame R relative to frame 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), Equation (2.18) simplifies to

$${}^U P = {}^U T_R \times {}^R P. \quad (2.21)$$

As you notice, canceling the R 's gives the coordinates of point P relative to U . The same notation will be used throughout this book to relate to multiple transformations.

Example 2.5

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 will be

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\theta & -S\theta \\ 0 & S\theta & C\theta \end{bmatrix} \times \begin{bmatrix} P_u \\ P_o \\ P_a \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ -4 \\ 3 \end{bmatrix}$$

As you notice in Figure 2.12, the coordinates of point P relative to the reference frame after rotation are $2, -4, 3$, as obtained by the preceding transformation.

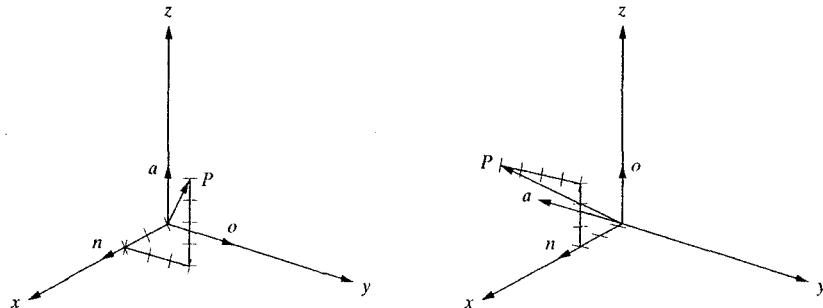


Figure 2.12 Rotation of a frame relative to the reference frame.

2.5.3 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, one may rotate a frame about the x -axis, translate about the x -, y -, and z -axes, and then rotate about the y -axis in order to accomplish the transformation that is needed. As we will see later, this order is very important, and if the order of two successive transformations is changed, the result may be completely different.

To see how combined transformations are handled, let's assume that a frame $(\bar{n}, \bar{o}, \bar{a})$ is subjected to the following three successive transformations relative to the reference frame (x, y, z) :

- (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 $(\bar{n}, \bar{o}, \bar{a})$ rotates or translates relative to the reference frame, the point P within the frame moves as well, 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

$$P_{1,xyz} = \text{Rot}(x, \alpha) \times P_{noa}, \quad (2.22)$$

where $P_{1,xyz}$ 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 will be:

$$P_{2,xyz} = \text{Trans}(l_1, l_2, l_3) \times P_{1,xyz} = \text{Trans}(l_1, l_2, l_3) \times \text{Rot}(x, \alpha) \times P_{noa}.$$

Similarly, after the third transformation, the coordinates of the point relative to the reference frame will be

$$P_{xyz} = P_{3,xyz} = \text{Rot}(y, \beta) \times P_{2,xyz} = \text{Rot}(y, \beta) \times \text{Trans}(l_1, l_2, l_3) \times \text{Rot}(x, \alpha) \times P_{noa}$$

As you see, the coordinates of the point relative to the reference frame at the conclusion of each transformation is found by premultiplying the coordinates of the point by each transformation matrix. Of course, as shown in Appendix A, the order of matrices cannot be changed. Please also notice that for each transformation relative to the reference frame, the matrix is premultiplied. Thus, the order of matrices *written* is the opposite of the order of transformations performed.

Example 2.6

A point $P(7,3,2)^T$ is attached to a frame $(\bar{n}, \bar{o}, \bar{a})$ and is subjected to the transformations described next. Find the coordinates of the point relative to the reference frame at the conclusion of 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 $[4, -3, 7]$.

Solution The matrix equation representing the transformation is

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

$$\begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 7 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 10 \\ 1 \end{bmatrix}$$

As you notice, the first transformation of 90° about the z -axis rotates the $(\bar{n}, \bar{o}, \bar{a})$ frame as shown in Figure 2.13, followed by the second rotation about the y -axis, followed by the translation relative to the reference frame x -, y -, z -axes. The point P in the frame can then be found relative to the \bar{n} -, \bar{o} -, \bar{a} -axes, as shown. The final coordinates of the point can be traced on the x -, y -, z -axes to be $4 + 2 = 6$, $-3 + 7 = 4$, and $7 + 3 = 10$. Please make sure that you follow this graphically.

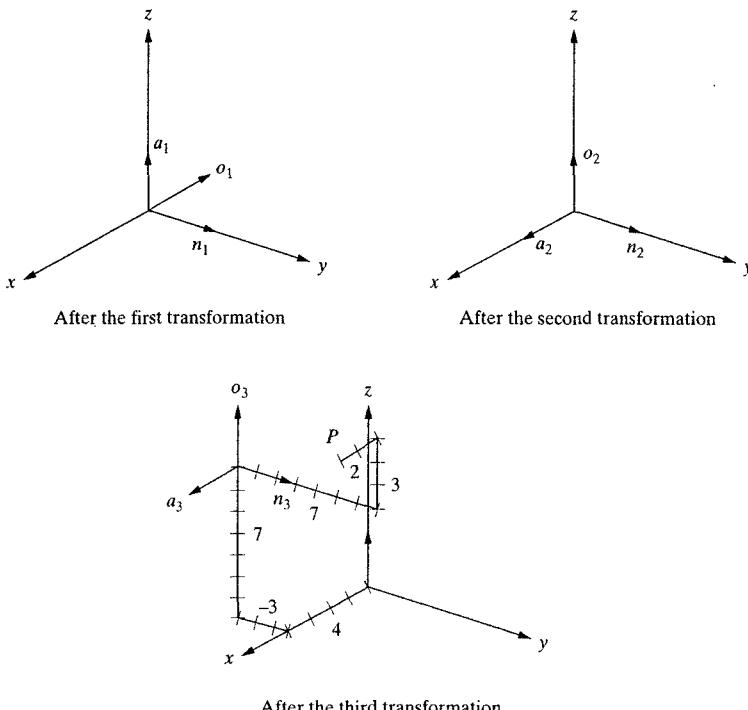


Figure 2.13 Effects of three successive transformations.

Example 2.7

In this case, assume that the same point $P(7,3,2)^T$, attached to a frame $(\bar{n}, \bar{o}, \bar{a})$, is subjected to the same transformations, but that the transformations are performed in a different order, as shown. Find the coordinates of the point relative to the reference frame at the conclusion of transformations:

- (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.

Solution The matrix equation representing the transformation is

$$P_{sys} = \text{Rot}(y, 90) \text{Trans}(4, -3, 7) \text{Rot}(z, 90) P_{moa} =$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 7 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 9 \\ 4 \\ -1 \\ 1 \end{bmatrix}$$

As you see, although the transformations are exactly the same as in Example 2.6, since the order of transformations is changed, the final coordinates of the point

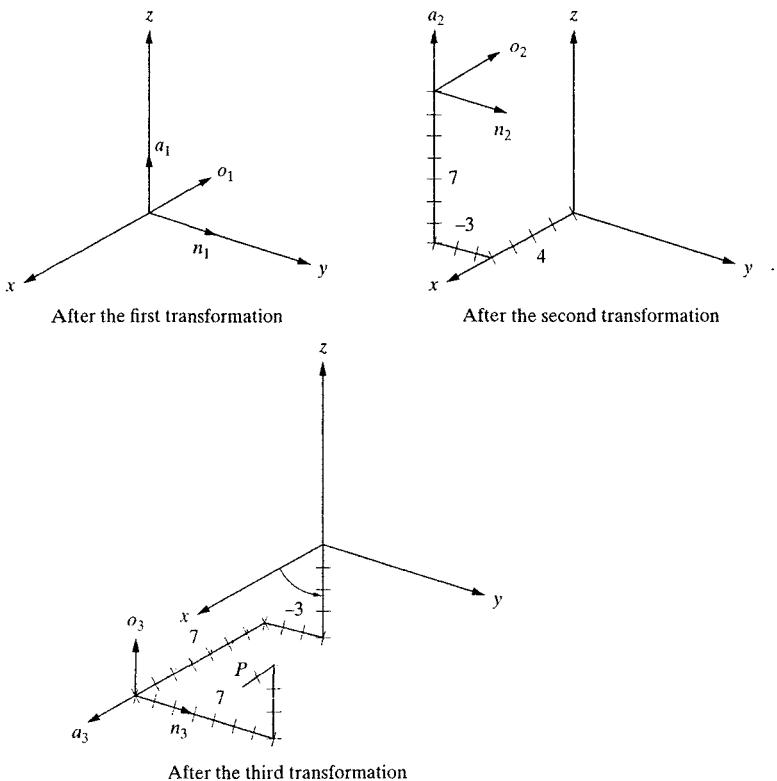


Figure 2.14 Changing the order of transformations will change the final result.

are completely different from the previous example. This can clearly be demonstrated graphically as in Figure 2.14. In this case you see that although the first transformation creates exactly the same change in the frame, the second transformation's result is very different, as the translation relative to the reference frame axes will move the rotating $(\bar{n}, \bar{o}, \bar{a})$ frame outwardly. As a result of the third transformation, this frame will rotate about the reference frame y -axis, thus rotating downwardly. The location of point P , attached to the frame is also shown.

Please verify that the coordinates of this point relative to the reference frame are $7 + 2 = 9$, $-3 + 7 = 4$, and $-4 + 3 = -1$, which is the same as the analytical result.

2.5.4 Transformations Relative to the Rotating 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 in fact possible to make transformations relative to the axes of a moving or current frame. This means that, for example, a rota-

tion of 90° may be made relative to the \bar{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 attached to the current frame relative to the reference frame, the transformation matrix is postmultiplied instead. Please 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 postmultiplied.

Example 2.8

Assume that the same point as in Example 2.7 is now subjected to the same transformations, but all relative to the current moving frame, as listed next. Find the coordinates of the point relative to the reference frame after transformations are completed:

- (1) A rotation of 90° about the \bar{a} -axis,
- (2) Then a translation of $[4, -3, 7]$ along $\bar{n}, \bar{o}, \bar{a}$
- (3) Followed by a rotation of 90° about the \bar{o} -axis.

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

$$P_{nzc} = \text{Rot}(\bar{a}, 90) \text{Trans}(4, -3, 7) \text{Rot}(\bar{o}, 90) P_{noa} =$$

$$\begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 7 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ 0 \\ 1 \end{bmatrix}.$$

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 2.15 shows the results graphically. Please notice how the transformations are accomplished relative to the current frames.

Please notice how the 7,3,2 coordinates of point P in the current frame will result in 0,6,0 coordinates relative to the reference frame.

Example 2.9

A frame B was rotated about the x -axis 90° ; it was then translated about the current a -axis 3 inches before being rotated about the z -axis 90° . Finally, it was translated about current o -axis 5 inches.

- (a) Write an equation describing the motions.
- (b) Find the final location of a point $P(1,5,4)$ attached to the frame relative to the reference frame.

For motion 1, abutment on wall must be avoided.

Solution In this case, motions alternate relative to the reference frame and current frame.

- (a) Pre- or postmultiplying each motion's matrix accordingly, we get

$${}^v T_B = \text{Rot}(z, 90) \text{Rot}(n, 90) \text{Trans}(0, 0, 3) \text{Trans}(0, 5, 0).$$

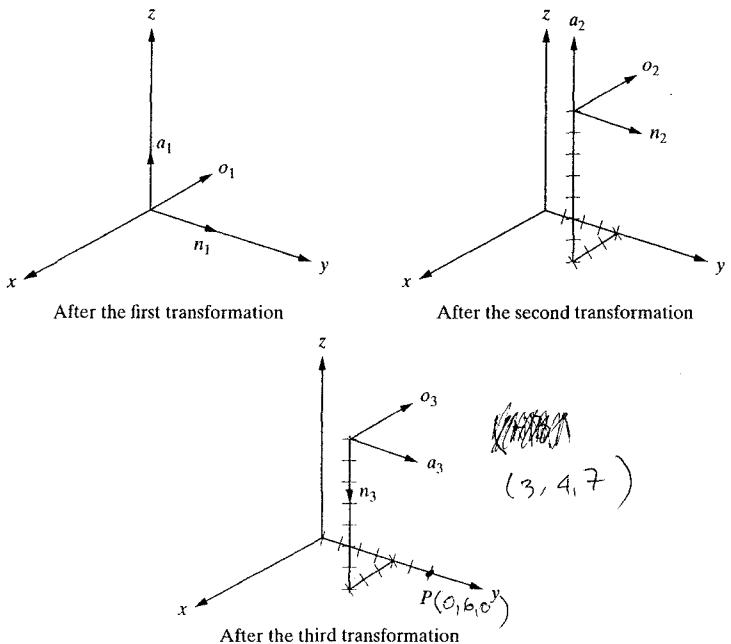


Figure 2.15 Transformations relative to the current frames.

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

$${}^uP = {}^uT_B \times {}^B P = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \\ 4 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 7 \\ 1 \\ 10 \\ 1 \end{bmatrix}$$

2.6 INVERSE OF TRANSFORMATION MATRICES

As was mentioned earlier, there are many situations where the inverse of a matrix will be needed in robotic analysis. One situation where transformation matrices may be involved can be seen in the next example. Suppose that the robot in Figure 2.16 is to be moved towards 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 also described by frame P . The location of the point where the hole will be

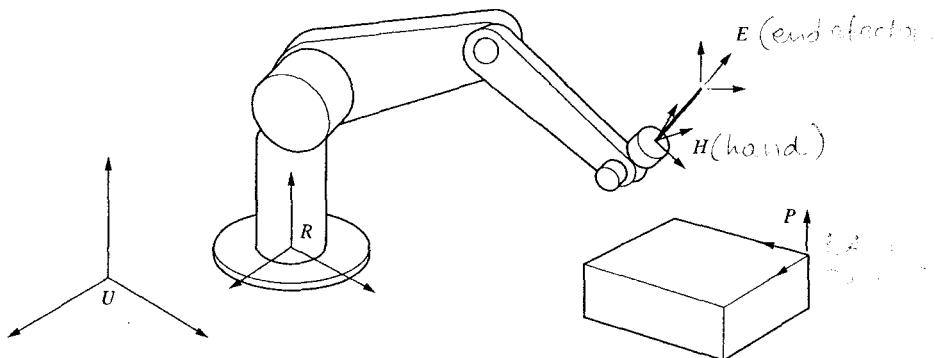


Figure 2.16 The Universe, robot, hand, part, and end effector frames.

drilled can be related to the reference frame U through two independent paths: one through the part and one through the robot. Thus, we can write

$${}^U T_E = {}^U T_R {}^R T_H {}^H T_E = {}^U T_P {}^P T_E, \quad (2.24)$$

which means that the location of point E on the part can be achieved by moving from U to P , and from P to E , or it can alternatively be achieved by a transformations from U to R , from R to H , and from H to E .

In reality, the transformation ${}^U T_R$, or the transformation of frame R relative to the U (Universe reference frame) 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 work cell, the location of the robot's base will be 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 will be known, since 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 that we need to find out what the robot's joint variables (the angle of the revolute joints and the length of the prismatic joints of the robot) must be in order to place the end effector at the hole for drilling. As you see, it is necessary to calculate this transformation, which will tell us what needs to be accomplished. The transformation will later be 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 postmultiply by inverses of appropriate matrices to eliminate them. As a result, we will have

$$({}^U T_R)^{-1} ({}^U T_R {}^R T_H {}^H T_E) ({}^H T_E)^{-1} = ({}^U T_R)^{-1} ({}^U T_P {}^P T_E) ({}^P T_E)^{-1}, \quad (2.25)$$

or since $({}^U T_R)^{-1}({}^U T_R) = 1$, and $({}^H T_E)({}^H T_E)^{-1} = 1$, the left side of Equation (2.25) simplifies to ${}^R T_H$, and we get

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

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

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

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

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

$$\text{Rot}(x, \theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\theta & -S\theta \\ 0 & S\theta & C\theta \end{bmatrix}. \quad (2.28)$$

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

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

Applying the process to the rotation matrix, we get

$$\det = 1(C^2\theta + S^2\theta) + 0 = 1,$$

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

Now calculate each minor. As an example, the minor for the 2,2 element will be $C\theta - 0 = C\theta$, the minor for 1,1 element will be $C^2\theta + S^2\theta = 1$, etc. As you notice, the minor for each element will be the same as the element itself. Thus,

$$\text{Rot}(x, \theta)^T_{\text{minor}} = \text{Rot}(x, \theta)^T.$$

Since the determinant of the original rotation matrix is unity, dividing the $\text{Rot}(x, \theta)^T_{\text{minor}}$ matrix by the determinant will yield the same result. Thus, the inverse of a rotation matrix about the x -axis is the same as its transpose, or

$$\text{Rot}(x, \theta)^{-1} = \text{Rot}(x, \theta)^T. \quad (2.29)$$

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. Thus, all we need to do to calculate the inverse of a rotation matrix is to transpose it. Please verify that rotation matrices about the y - and z -axes are also unitary in nature.

Please beware that only rotation matrices are unitary. If a matrix is not a simple rotation matrix, it may not be unitary.

The preceding result is also true only for a simple 3×3 rotation matrix without representation of a location. For a homogenous 4×4 transformation matrix, it can be shown that the matrix inverse 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 vector \bar{P} with each of the $\bar{n}, \bar{o}, \bar{a}$ vectors as follows:

$$T = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\bar{P} \cdot \bar{n} \\ o_x & o_y & o_z & -\bar{P} \cdot \bar{o} \\ a_x & a_y & a_z & -\bar{P} \cdot \bar{a} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.30)$$

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) are not affected. This is very helpful, since we will need to calculate inverses of transformation matrices, but directly calculating inverse of a 4×4 matrix is a lengthy process.

Example 2.10

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

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

As you notice, since the position vector of the matrix is zero, its dot product with the $\bar{n}, \bar{o}, \bar{a}$ vectors is also zero.

Example 2.11

Calculate the inverse of the following transformation matrix:

$$T = \begin{bmatrix} .5 & 0 & .866 & 3 \\ .866 & 0 & -.5 & 2 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution Based on the preceding calculations, the inverse of the transformation is

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

You may want to verify that $T T^{-1}$ will be an identity matrix.

Example 2.12

In a robotic setup, a camera is attached to the fifth link of a robot with six degrees of freedom. The camera observes an object and determines its frame relative to the camera's frame. Using the following information, determine the necessary motion the end effector has to make to get to the object:

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

Solution Referring to Equation (2.24), we can write a similar equation that relates the different transformations and frames together as

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

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

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

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

Substituting the matrices and the inverses in the foregoing equation results in:

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

2.7 FORWARD AND INVERSE KINEMATICS OF ROBOTS

Suppose that we have a robot whose configuration is known. This means that all the link lengths and joint angles of the robot 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, one can calculate where the robot is at any instant. However, if one desires to place the hand of the robot at a desired location and orientation, one has to know how much each link length or joint angle of the robot 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, what is more important is the inverse kinematic equations, since the robot controller will calculate the joint values using these equations, and it will run the robot to the desired position and orientation. We will first develop the forward kinematic equations of robots, and then, using these equations, we will calculate the inverse kinematic equations.

For forward kinematics, we will have to develop a set of equations that relate to the particular configuration of a robot (the way it is put together) such that by substituting the joint and link variables in these equations, we may calculate the position and orientation of the robot. These equations will then be used to derive the inverse kinematic equations.

You may recall from Chapter 1 that in order to position and orientate a rigid body in space, we attach a frame to the body and then describe the position of the origin of the frame and the orientation of its three axes. This requires a total of six degrees of freedom, or, alternatively, 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 the robot in space, we will attach a frame to it and will define the position and orientation of the hand frame of the robot. The means by which the robot accomplishes this determines what the forward kinematic equations are. In other words, depending on the configuration of the links and joints

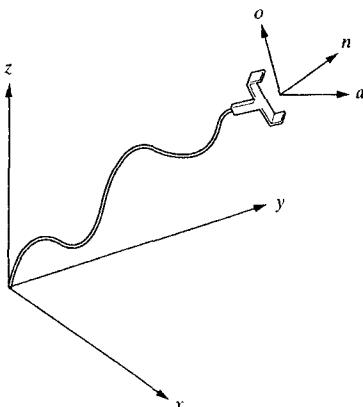


Figure 2.17 The hand frame of the robot relative to the reference frame.

of the robot, a particular set of equations will relate the hand frame of the robot to the reference frame. Figure 2.17 shows a hand frame, the reference frame, and their relative position and orientation. The undefined connection between the two frames is related to the configuration of the robot. Of course, there may be 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 will analyze the position and orientation issues separately. First, we will develop the position equations; then we will do the same for orientation. Later, we will combine the two for a complete set of equations. Finally, we will see about the use of the Denavit–Hartenberg representation, which can model any robot configuration.

2.7.1 Forward and Inverse Kinematic Equations for Position

In this section, we will 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 three degrees of freedom and thus can be completely defined by three pieces of information. As a result, the position of the origin of the frame may be defined in any customary coordinates, and the positioning of the robot may be accomplished through motions related to any customary coordinate frames. For example, one may position a point in space based on Cartesian coordinates, which means that there will be three linear movements relative to the \bar{x} , \bar{y} , and \bar{z} -axes. Alternatively, it may be accomplished through spherical coordinates, which means that there will be one linear motion and two rotary motions to accomplish the position. The following possibilities will be discussed:

- (a) Cartesian (gantry, rectangular) coordinates.
- (b) Cylindrical coordinates.
- (c) Spherical coordinates.
- (d) Articulated (anthropomorphic, or all-revolute) coordinates.

2.7.1(a) Cartesian (Gantry, Rectangular) Coordinates

In this case, there will be three linear movements along the three major x , y , z -axes. In this type of a 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 three axes (Figure 2.18). A gantry robot is basically a Cartesian coordinate robot, except that the robot is usually attached to a rectangular frame upside down. The IBM 7565 robot is a gantry Cartesian robot.

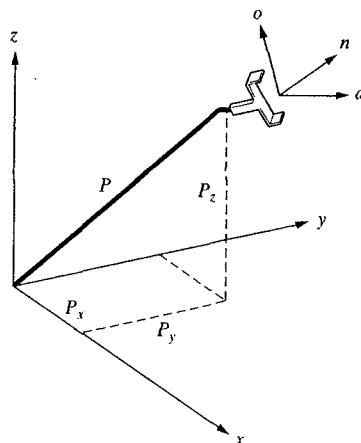


Figure 2.18 Cartesian coordinates.

Of course, since there are no rotations, the transformation matrix representing this motion to point P is a simple translation transformation matrix, as shown next. Please note that here we are only referring to the position of the origin of the frame, and not its orientation. The transformation matrix representing the forward kinematic equation of the position of the hand of the robot in a Cartesian coordinate system will be

$${}^R T_P = T_{\text{cart}} = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.31)$$

where ${}^R T_P$ is the transformation between the reference frame and the origin of the hand P , and T_{cart} denotes Cartesian transformation matrix. For inverse kinematic solution, simply set the desired position equal to P .

Example 2.13

It is desired to position the origin of the hand frame of a Cartesian robot at point $P = [3, 4, 7]^T$. 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 Equation (2.31), with the desired position will yield the following result:

$${}^R T_P = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{or} \quad P_x = 3, P_y = 4, P_z = 7.$$

2.7.1(b) 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 2.19. Since these transformations are all relative to the axes of the universe reference frame, the total transformation caused by the three transformations that relates the origin of the hand frame to the reference frame can be found by premultiplying by each matrix as follows:

$${}^R T_P = T_{\text{cyl}}(r, \alpha, l) = \text{Trans}(0, 0, l) \text{Rot}(z, \alpha) \text{Trans}(r, 0, 0), \quad (2.32)$$

$$\begin{aligned} {}^R T_P &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} C\alpha & -S\alpha & 0 & 0 \\ S\alpha & C\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & r \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ {}^R T_P &= T_{\text{cyl}} = \begin{bmatrix} C\alpha & -S\alpha & 0 & rC\alpha \\ S\alpha & C\alpha & 0 & rS\alpha \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \quad (2.33)$$

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

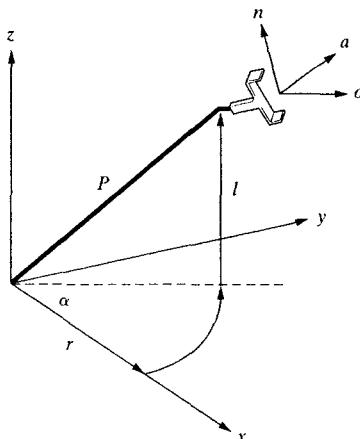


Figure 2.19 Cylindrical Coordinates.

the origin of the frame, or the last column. Obviously, in cylindrical coordinate movements, due to the rotation α about the z -axis, the orientation of the moving frame will change. This orientation change will be discussed later.

One may, in fact, “unrotate” the frame back to being parallel to the original reference frame by rotating the $\bar{n}, \bar{o}, \bar{a}$ frame about the a -axis an angle of $-\alpha$, which is equivalent of post-multiplying the cylindrical coordinate matrix by a rotation matrix of $\text{Rot}(\bar{a}, -\alpha)$. As a result, the frame will be at the same location, but will be parallel to the reference frame again, as follows:

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

As you can see, the location of the origin of the moving frame has not changed, but it was “unrotated” back to being parallel to the reference frame. Please notice that the last rotation was performed about the local a -axis in order to not cause any change in the location of the frame, but only in its orientation.

Example 2.14 (Robotic hand example for unrotated A-frame)

Suppose that we desire to place the origin of the hand frame of a cylindrical robot at $[3, 4, 7]^T$. Calculate the joint variables of the robot.

Solution Setting the components of the location of the origin of the frame from the T_{cyl} matrix of Equation (2.33) to the desired values, we get

$$\begin{aligned} l &= 7, \\ rC\alpha &= 3, \end{aligned}$$

$$rS\alpha = 4, \text{ and thus } \tan \alpha = \frac{4}{3} \text{ and } \alpha = 53.1^\circ.$$

Substituting α into either equation will yield $r = 5$. The final answer is

$$r = 5 \text{ units, } \alpha = 53.1^\circ, \text{ and } l = 7 \text{ units.}$$

Note: As discussed in Appendix A, it is necessary to ensure that the angles calculated in robot kinematics are in correct quadrants. In this example, you notice that since $rC\alpha$ and $rS\alpha$ are both positive and that the length r is always positive, that the sin and cos are also both positive. Thus, the angle α is in the first quadrant and is correctly 53.1° .

2.7.1(c) Spherical Coordinates

Spherical coordinate systems consist of one linear motion and two rotations. The sequence is a translation of r along the z -axis, a rotation of β about the y -axis, and a

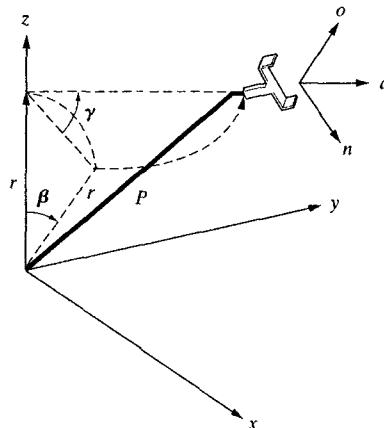


Figure 2.20 Spherical coordinates.

rotation of γ about the z -axis as shown in Figure 2.20. Since these transformations are all relative to the axes of the Universe reference frame, the total transformation caused by these three transformations, which relates the origin of the hand frame to the reference frame, can be found by premultiplying by each matrix as follows:

$${}^R T_P = T_{\text{sph}}(r, \beta, \gamma) = \text{Rot}(z, \gamma) \text{Rot}(y, \beta) \text{Trans}(0, 0, r) \quad (2.34)$$

$$\begin{aligned} {}^R T_P &= \begin{bmatrix} C\gamma & -S\gamma & 0 & 0 \\ S\gamma & C\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} C\beta & 0 & S\beta & 0 \\ 0 & 1 & 0 & 0 \\ -S\beta & 0 & C\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\quad \text{N} \text{X} \quad \text{o} \text{Y} \quad \text{o} \text{Z} \quad \text{P} \\ {}^R T_P &= T_{\text{sph}} = \begin{bmatrix} C\beta \cdot C\gamma & -S\gamma & S\beta \cdot C\gamma & rS\beta \cdot C\gamma \\ C\beta \cdot S\gamma & C\gamma & S\beta \cdot S\gamma & rS\beta \cdot S\gamma \\ -S\beta & 0 & C\beta & rC\beta \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \quad (2.35)$$

Once again, the first three columns represent the orientation of the frame after this series of transformations, while the last column is the position of the origin. We will further discuss the orientation part of the matrix later.

Here, too, one may “unrotate” the final frame to make it parallel to the reference frame. This exercise is left for you 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 2.15

Suppose that we now desire to place the origin of the hand of a spherical robot at $[3, 4, 7]^T$. Calculate the joint variables of the robot.

Solution Setting the components of the location of the origin of the frame from $T_{\text{sp}h}$ matrix of Equation (2.35) to the desired values, we get

$$rS\beta C\gamma = 3,$$

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

$$rC\beta = 7.$$

From the third equation, we determine that $C\beta$ is positive, but there is no such information about $S\beta$. Dividing the first two equations by each other, we get the following double results. Please notice that there are two possible solutions, because we do not know what the actual sign of $S\beta$ is. As a result, the following approach presents two possible solutions, and we will have to check the final results later to ensure that they are correct:

$$\begin{array}{llll} \tan \gamma = \frac{4}{3} & \rightarrow & \gamma = 53.1^\circ, & \text{or} \quad 233.1^\circ, \\ \text{then} & & S\gamma = 0.8, & \text{or} \quad -0.8, \\ \text{and} & & C\gamma = 0.6, & \text{or} \quad -0.6, \\ \text{and} & & rS\beta = \frac{3}{0.6} = 5, & \text{or} \quad -5, \\ \text{and since} & & rC\beta = 7, \beta = 35.5^\circ, & \text{or} \quad -35.5^\circ, \\ \text{and} & & r = 8.6. & \end{array}$$

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 three dimensions you will 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, the two foregoing answers will 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 may not specify any orientation for a three-degree-of-freedom robot, we cannot determine which of the two answers relates to a specific orientation anyway.

2.7.1(d) Articulated Coordinates

Articulated coordinates consist of three rotations, as shown in Figure 2.21. We will develop the matrix representation for this later, when we discuss the Denavit–Hartenberg representation.

2.7.2 Forward and Inverse Kinematic Equations for Orientation

Suppose that the moving frame attached to the hand of the robot has already moved to a desired position, but is still parallel to the reference frame, or that it is in an ori-

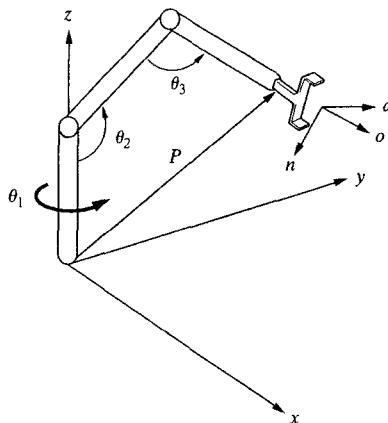


Figure 2.21 Articulated Coordinates.

entation other than what is desired. The next step will be to rotate the frame appropriately in order to achieve a desired orientation without changing its 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 will consider the following three common configurations:

- (a) Roll, Pitch, Yaw (RPY) angles.
- (b) Euler angles
- (c) Articulated joints

2.7.2(a) Roll, Pitch, Yaw (RPY) Angles

This is a sequence of three rotations about current $\bar{a}, \bar{o}, \bar{n}$ axes respectively, which will orientate the hand of the robot to a desired orientation. The assumption here is that the current frame is parallel to the reference frame and thus its orientation is the same as the reference frame before the RPY movements. If the current moving frame is not parallel to the reference frame, then the final orientation of the robot's hand will be a combination of the previous orientation, postmultiplied by the RPY.

It is very important to realize that since we do not want to cause any change in the position of the origin of the moving frame (we have already placed it at the desired location and only want to rotate it to the desired orientation), the movements relating to RPY rotations are relative to the current moving axes. Otherwise, as we saw before, the moving frame will change position. Thus, all matrices relating to the orientation change due to RPY (as well as other rotations) will be postmultiplied.

Referring to Figure 2.22, we see that the RPY sequence of rotations consists of the following:

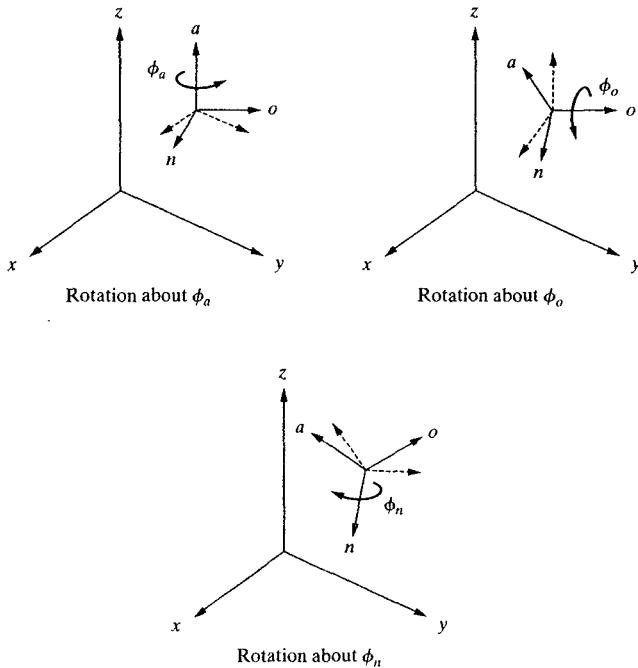


Figure 2.22 RPY rotations about the current axes.

Rotation of ϕ_a about the \bar{a} -axis (z-axis of the moving frame) called Roll,
 Rotation of ϕ_o about the \bar{o} -axis (y-axis of the moving frame) called Pitch,
 Rotation of ϕ_n about the \bar{n} -axis (x-axis of the moving frame) called Yaw.

The matrix representing the RPY orientation change will be

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

$$\begin{bmatrix} C\phi_a C\phi_o & C\phi_a S\phi_o S\phi_n - S\phi_a C\phi_n & C\phi_a S\phi_o C\phi_n + S\phi_a S\phi_n & 0 \\ S\phi_a C\phi_o & S\phi_a S\phi_o S\phi_n + C\phi_a C\phi_n & S\phi_a S\phi_o C\phi_n - C\phi_a S\phi_n & 0 \\ -S\phi_o & C\phi_o S\phi_n & C\phi_o C\phi_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.36)$$

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

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

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 them. To solve for these sines and cosines, we will have to decouple these angles. To do this, we will premultiply both sides of Equation by the inverse of $\text{Rot}(a, \phi_a)$:

$$\text{Rot}(a, \phi_a)^{-1} \text{RPY}(\phi_a, \phi_o, \phi_n) = \text{Rot}(o, \phi_o) \text{Rot}(n, \phi_n). \quad (2.37)$$

Assuming that the final desired orientation achieved by RPY is represented by the $(\bar{n}, \bar{o}, \bar{a})$ matrix, we have

$$\text{Rot}(a, \phi_a)^{-1} \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \text{Rot}(o, \phi_o) \text{Rot}(n, \phi_n). \quad (2.38)$$

Multiplying the matrices, we get

$$\begin{bmatrix} n_x C\phi_a + n_y S\phi_a & o_x C\phi_a + o_y S\phi_a & a_x C\phi_a + a_y S\phi_a & 0 \\ n_y C\phi_a - n_x S\phi_a & o_y C\phi_a - o_x S\phi_a & a_y C\phi_a - a_x S\phi_a & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\phi_o & S\phi_o S\phi_n & S\phi_o C\phi_n & 0 \\ 0 & C\phi_n & -S\phi_n & 0 \\ -S\phi_o & C\phi_o S\phi_n & C\phi_o C\phi_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.39)$$

Please remember that the $\bar{n}, \bar{o}, \bar{a}$ components in Equation (2.38) represent the final desired values, which are normally given or known. The values of the RPY angles are the unknown variables.

Equating the different elements of the right- and the left-hand sides of Equation (2.39) will result in the following (please refer to Appendix A for explanation of ATAN2 function):

From the 2,1 elements, we get

$$n_y C\phi_a - n_x S\phi_a = 0 \rightarrow \phi_a = \text{ATAN2}(n_y, n_x) \quad \text{and} \quad \phi_o = \text{ATAN2}(-n_y, -n_x). \quad (2.40)$$

From the 3,1 and 1,1 elements, we get

$$\begin{aligned} S\phi_o &= -n_z, \\ C\phi_o &= n_x C\phi_a + n_y S\phi_a \rightarrow \phi_o = \text{ATAN2}(-n_z, (n_x C\phi_a + n_y S\phi_a)). \end{aligned} \quad (2.41)$$

Finally, from 2,2 and 2,3, elements, we get

$$\begin{aligned}
 C\phi_a &= o_y C\phi_a - o_x S\phi_a, \\
 S\phi_a &= -a_y C\phi_a + a_x S\phi_a \rightarrow \\
 \phi_a &= \text{ATAN2}[-a_y C\phi_a + a_x S\phi_a, (o_y C\phi_a - o_x S\phi_a)]. \quad (2.42)
 \end{aligned}$$

Example 2.16

The desired final position and orientation of the hand of a Cartesian–RPY robot is given next. Find the necessary roll, pitch, and yaw angles and displacements:

$${}^R T_p = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.354 & -0.674 & 0.649 & 4.33 \\ 0.505 & 0.722 & 0.475 & 2.50 \\ -0.788 & 0.160 & 0.595 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Solution From the preceding equations, we find two sets of answers:

$$\begin{aligned}
 \phi_a &= \text{ATAN2}(n_y, n_x) = \text{ATAN2}(0.505, 0.354) = 55^\circ, \text{ or } 235^\circ, \\
 \phi_o &= \text{ATAN2}(-n_z, (n_x C\phi_a + n_y S\phi_a)) = \text{ATAN2}(0.788, 0.616) = 52^\circ, \text{ or } 128^\circ, \\
 \phi_n &= \text{ATAN2}((-a_y C\phi_a + a_x S\phi_a), (o_y C\phi_a - o_x S\phi_a)) \\
 &\quad = \text{ATAN2}(0.259, 0.966) = 15^\circ, \text{ or } 195^\circ, \\
 p_x &= 4.33, \quad p_y = 2.5, \quad p_z = 8 \text{ units.}
 \end{aligned}$$

Example 2.17

For the same position and orientation as in Example 2.16, find all necessary joint variables if the robot is cylindrical–RPY.

Solution In this case, we will use

$${}^R T_p = \begin{bmatrix} 0.354 & -0.674 & 0.649 & 4.33 \\ 0.505 & 0.722 & 0.475 & 2.50 \\ -0.788 & 0.160 & 0.595 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} = T_{\text{cyl}}(r, \alpha, l) \times \text{RPY}(\phi_a, \phi_o, \phi_n).$$

The right-hand side of this equation now involves four angles that are coupled and must be decoupled as previously shown. However, since the rotation of α about z -axis for the cylindrical coordinates does 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 angle we found for ϕ_a is in fact the summation of $\phi_a + \alpha$. (See Figure 2.23.) Using the position information given, the solution of Example 2.16, and referring to Equation (2.33), we will get

$$\begin{aligned}
 rC\alpha &= 4.33, \quad rS\alpha = 2.5 \quad \rightarrow \quad \alpha = 30^\circ, \\
 \phi_a + \alpha &= 55^\circ \quad \rightarrow \quad \phi_a = 25^\circ, \\
 S\alpha &= 0.5 \quad \rightarrow \quad r = 5,
 \end{aligned}$$

Please remember that the $\bar{n}, \bar{o}, \bar{a}$ components in Equation (2.44) represent the final desired values, which are normally given or known. The values of the Euler angles are the unknown variables.

Equating the different elements of the right- and the left-hand sides of Equation (2.45) will result in the following:

From the 2,3 elements, we get

$$-a_x S\phi + a_y C\phi = 0 \rightarrow \phi = \text{ATAN2}(a_y, a_x), \quad \text{or} \quad \phi = \text{ATAN2}(-a_y, -a_x). \quad (2.46)$$

With ϕ evaluated, all the elements of the left-hand side of Equation (2.45) are known. From the 2,1 and 2,2 elements, we get

$$\begin{aligned} S\psi &= -n_x S\phi + n_y C\phi, \\ C\psi &= -o_x S\phi + o_y C\phi \rightarrow \psi = \text{ATAN2}(-n_x S\phi + n_y C\phi, -o_x S\phi + o_y C\phi). \end{aligned} \quad (2.47)$$

Finally, from 1,3 and 3,3 elements, we get

$$\begin{aligned} S\theta &= a_x C\phi + a_y S\phi, \\ C\theta &= a_z \rightarrow \theta = \text{ATAN2}(a_x C\phi + a_y S\phi, a_z). \end{aligned} \quad (2.48)$$

Example 2.18

The desired final orientation of the hand of a Cartesian–Euler robot is given. Find the necessary Euler angles:

$${}^R T_H = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.579 & -0.548 & -0.604 & 5 \\ 0.540 & 0.813 & -0.220 & 7 \\ 0.611 & -0.199 & 0.766 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Solution From the foregoing equations, we find

$$\phi = \text{ATAN2}(a_y, a_x) = \text{ATAN2}(-0.220, -0.604) = 20^\circ \text{ or } 200^\circ.$$

Realizing that both the sines and cosines of 20° and 200° can be used for the remainder, we find that

$$\psi = \text{ATAN2}(-n_x S\phi + n_y C\phi, -o_x S\phi + o_y C\phi) = (0.31, 0.952) = 18^\circ, \text{ or } 198^\circ,$$

$$\theta = \text{ATAN2}(a_x C\phi + a_y S\phi, a_z) = \text{ATAN2}(-0.643, 0.766) = -40^\circ, \text{ or } 40^\circ.$$

2.7.2(c) Articulated Joints

Articulated joints consist of three rotations other than the ones just presented, which are based on the design of the joints. As we did in section 2.7.1(d), we will develop the matrix representing articulated joints later, when we discuss the Denavit–Hartenberg representation.

2.7.3 Forward and Inverse Kinematic Equations for Position and Orientation

The matrix representing the final location and orientation of the robot is a combination of the preceding equations, depending on which coordinates are used. Suppose that a robot is made of a Cartesian and an RPY set of joints. Then the location and the final orientation of the frame relative to the reference frame will be the product of the two matrices representing the Cartesian position change and the RPY. The robot may be represented by

$${}^R T_H = {}^{\text{Position}} T_{\text{cart}}(P_x, P_y, P_z) \times {}^{\text{Orientation}} \text{RPY}(\phi_a, \phi_o, \phi_n). \quad (2.49)$$

If the robot is designed based on spherical coordinates for positioning and Euler angles for orientation, then the final answer will be the following equation, where the position is determined by the spherical coordinates, while the final orientation is effected by both the angles in the spherical coordinates, as well as the Euler angles:

$${}^R T_H = {}^{\text{Position}} T_{\text{sph}}(r, \beta, \gamma) \times {}^{\text{Orientation}} \text{Euler}(\phi, \theta, \psi). \quad (2.50)$$

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.

2.8 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 the standard way of representing robots and modeling their motions and thus is essential to learn. The Denavit–Hartenberg (D–H) model of representation is a very simple way of modeling robot links and joints that can be used for any robot configuration, regardless of its 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 are faster and more straightforward, the Denavit–Hartenberg representation has the added benefit that many techniques have been developed for use with its results [5,6,7,8,9], such as the calculation of Jacobians, force analysis, etc.

Let's assume that robots may be made of a succession of joints and links. The joints may be either prismatic (linear) or revolute (rotational), they may be in any order or sequence, and they may be in any plane. The links may also be of any length, including zero, may be twisted or bent, and may be in any plane. Thus, any general set of joints and links may create a robot that we would like to model and represent.

$$p_z = 8 \rightarrow l = 8.$$

$$\text{As in Example 2.16,} \rightarrow \phi_o = 52^\circ, \quad \phi_n = 15^\circ.$$

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

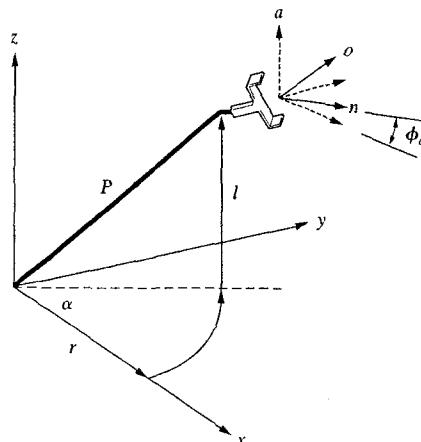


Figure 2.23 Cylindrical and RPY coordinates of Example 2.17.

2.7.2(b) Euler Angles

Euler angles are very similar to RPY, except that the last rotation is also about the current \bar{a} -axis. (See Figure 2.24.) We still need to make all rotations relative to the current axes to prevent any change in the position of the robot. Thus, the rotations representing the Euler angles are as follows:

- Rotation of ϕ about the \bar{a} -axis (z -axis of the moving frame) followed by,
- Rotation of θ about the \bar{o} -axis (y -axis of the moving frame) followed by,
- Rotation of ψ about the \bar{a} -axis (z -axis of the moving frame).

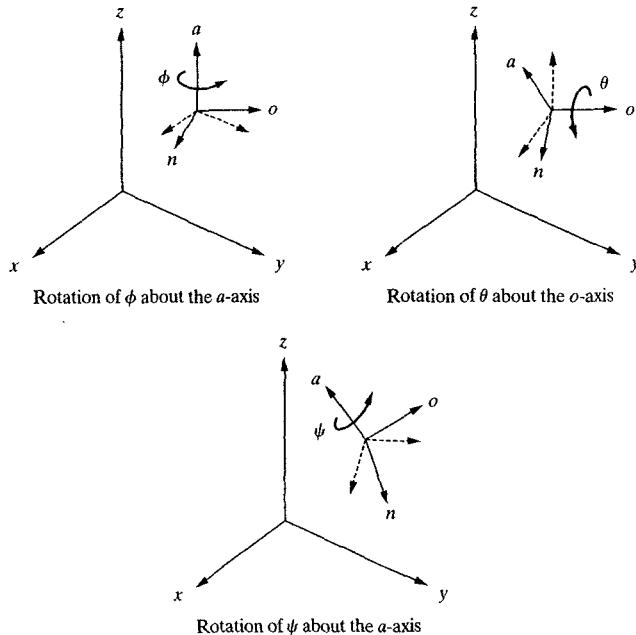
Then the matrix representing the Euler angles orientation change is

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

$$\begin{bmatrix} C\phi C\theta C\psi - S\phi S\psi & -C\phi C\theta S\psi - S\phi C\psi & C\phi S\theta & 0 \\ S\phi C\theta C\psi + C\phi S\psi & -S\phi C\theta S\psi + C\phi C\psi & S\phi S\theta & 0 \\ -S\theta C\psi & S\theta S\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.43)$$

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 can be found in a manner very similar to RPY. We will premultiply the two sides of the Euler equation by

**Figure 2.24** Euler rotations about the current axes.

$\text{Rot}^{-1}(a, \phi)$ to eliminate ϕ from one side. By equating the elements of the two sides to each other, we find the following equations (assuming that the final desired orientation achieved by Euler angles is represented by $(\bar{n}, \bar{o}, \bar{a})$ matrix

$$\text{Rot}^{-1}(a, \phi) \times \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\theta C\psi & -C\theta S\psi & S\theta & 0 \\ S\psi & C\psi & 0 & 0 \\ -S\theta C\psi & S\theta S\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.44)$$

or

$$\begin{bmatrix} n_x C\phi + n_y S\phi & o_x C\phi + o_y S\phi & a_x C\phi + a_y S\phi & 0 \\ -n_x S\phi + n_y C\phi & -o_x S\phi + o_y C\phi & -a_x S\phi + a_y C\phi & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\theta C\psi & -C\theta S\psi & S\theta & 0 \\ S\psi & C\psi & 0 & 0 \\ -S\theta C\psi & S\theta S\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.45)$$

To do this, we will need to assign a reference frame to each joint and, later, define 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 sections that follow, we will define the general procedure, based on Denavit–Hartenberg representation, to assign reference frames to each joint. Then we will define how a transformation between any two successive frames may be accomplished. Finally, we will write the total transformation matrix for the robot.

Imagine that a robot may be made of any number of links and joints in any form. Figure 2.25 represents three successive joints and two links. 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 joints in real robots. These joints may be revolute, prismatic, or both. Although in real robots it is customary to only have one degree of freedom joints, the joints in the figure represent one- or two-degree-of-freedom joints.

Figure 2.25(a) shows three joints. Each joint may rotate or translate. Let's assign joint number n to the first shown joint, $n+1$ to the second shown joint, and $n+2$ to the third shown joint. There may be other joints before or after these joints. 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 $n+2$.

To model the robot with D–H representation, the first thing we need to do is to assign a local reference frame for each and every joint. Thus, for each joint, we will have to assign a z -axis and an x -axis. 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 D–H 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 for the joint is along the direction of the linear movement. The index number for the z -axis of joint n (as well as the local reference frame for the joint) in each case is $n-1$. For example, the z -axis representing joint number $n+1$ is z_n . These simple rules will allow us to quickly assign z -axes to all joints. For revolute joints, the rotation about the z -axis (θ) will be the joint variable. For prismatic joint, the length of the link along the z -axis represented by d will be the joint variable.
- As you have noticed in Figure 2.25(a), in general, the joints may not necessarily be parallel or intersecting. As a result, in general, the z -axes are skew lines. There is always one line mutually perpendicular to any two skew lines, called the common normal, which has the shortest distance between the two skew lines. We always assign the x -axis of the local reference frames in the direction of the common normal. Thus, if a_n represents the common normal between z_{n-1} and z_n , the direction of x_n will be 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} . As you

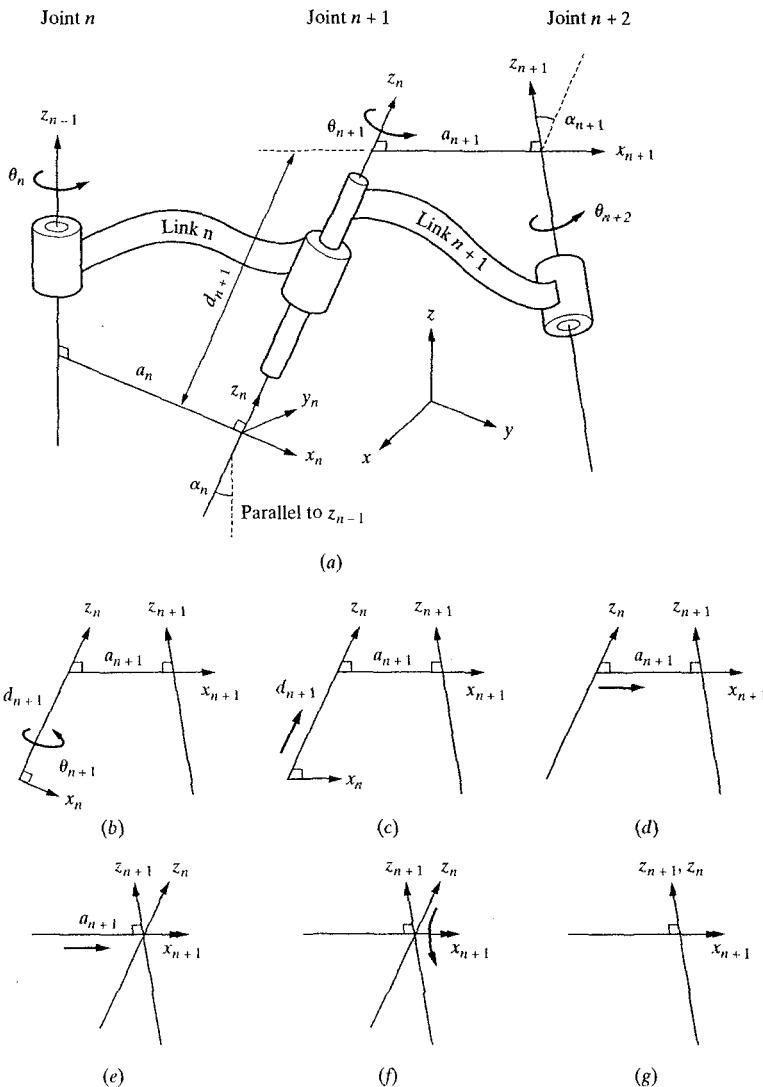


Figure 2.25 A D–H representation of a general-purpose joint–link combination.

notice, the common normal lines between successive joints are not necessarily intersecting or collinear. As a result, the location of the origins of two successive frames also may not be at the same location. Based on the preceding information and with the exception of the following special cases, we can assign coordinate frames to all joints:

- If two joint z -axes are parallel, there are an infinite number of common normals present. We will 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 will 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 and is equivalent of picking the direction of the cross product of the two z -axes. This also simplifies the model.

In Figure 2.25(a), θ represents a rotation about the z -axis, d represents the distance on the z -axis between two successive common normals, a represents the length of each common normal (also called joint offset), and α represents the angle between two successive z -axes (also called joint twist). Commonly, only θ and d are joint variables.

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

- (I) Rotate about the z_n -axis an angle of θ_{n+1} (Figures 2.25(a) and (b)). This will make x_n and x_{n+1} parallel to each other. 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 thus coplanar).
- (II) Translate along the z_n -axis a distance of d_{n+1} to make x_n and x_{n+1} colinear (Figure 2.25(c)). 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.
- (III) Translate along the x_n -axis a distance of a_{n+1} to bring the origins of x_n and x_{n+1} together (Figures 2.25(d) and (e)). At this point, the two origins of the two reference frames will be at the same location.
- (IV) Rotate z_n -axis about x_{n+1} -axis an angle of α_{n+1} to align z_n -axis with z_{n+1} -axis (Figure 2.25 (f)). At this point, frames n and $n+1$ will be exactly the same (Figure 2.25(g)), and we will have transformed from one frame to the next.

Doing exactly the same sequence of four movements between the $n+1$ and $n+2$ frames will transform one to the next, and by repeating this as necessary, we can transform between successive frames. Starting with the reference frame, we can transform to the base of the robot, then to the first joint, second joint, . . . , until the end effector. What is nice is that the foregoing sequence of movements remains the same between any two frames.

The matrix A representing the four movements is found by postmultiplying the four matrices representing the four movements. Since all transformations are relative to the current frame (i.e., they are measured and performed relative to the axes of the current local frame), all matrices are postmultiplied. The result is as follows:

$${}^nT_{n+1} = A_{n+1} = \text{Rot}(z, \theta_{n+1}) \times \text{Trans}(0,0,d_{n+1}) \times \text{Trans}(a_{n+1},0,0) \times \text{Rot}(x, \alpha_{n+1})$$

$$= \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1} & 0 & 0 \\ S\theta_{n+1} & C\theta_{n+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.51)$$

$$\times \begin{bmatrix} 1 & 0 & 0 & a_{n+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_{n+1} & -S\alpha_{n+1} & 0 \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_{n+1} = \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1}C\alpha_{n+1} & S\theta_{n+1}S\alpha_{n+1} & a_{n+1}C\theta_{n+1} \\ S\theta_{n+1} & C\theta_{n+1}C\alpha_{n+1} & -C\theta_{n+1}S\alpha_{n+1} & a_{n+1}S\theta_{n+1} \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.52)$$

For example, the transformation between joints two and three of a generic robot will simply be

$${}^2T_3 = A_3 = \begin{bmatrix} C\theta_3 & -S\theta_3C\alpha_3 & S\theta_3S\alpha_3 & a_3C\theta_3 \\ S\theta_3 & C\theta_3C\alpha_3 & -C\theta_3S\alpha_3 & a_3S\theta_3 \\ 0 & S\alpha_3 & C\alpha_3 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.53)$$

At the base of the robot, we can start with the first joint and transform to the second joint, then to the third, . . . , to the hand of the robot, and eventually to the end effector. Calling each transformation an A_{n+1} , we will have a number of A matrices that represent the transformations. The total transformation between the base of the robot and the hand will be

$${}^R T_H = {}^R T_1 {}^1 T_2 {}^2 T_3 \dots {}^{n-1} T_n = A_1 A_2 A_3 \dots A_n, \quad (2.54)$$

where n is the joint number. For a six-degree-of-freedom robot, there will be six A matrices.

To facilitate the calculation of the A matrices, we will form a table of joint and link parameters whereby the values representing each link and joint are determined from the schematic drawing of the robot, and are substituted into each A matrix. Table 2.1 can be used for this purpose.

In the next few examples, we will assign the necessary frames, fill out the parameters tables, and substitute the values into the A matrices. We start with a simple robot, but will consider more difficult robots later.

Example 2.19

For the simple robot shown in Figure 2.26, assign the necessary coordinate frames based on the D–H representation, and fill out the accompanying parameters table.

Solution In this example, for simplicity, we assume that joints 2, 3, and 4 are in the same plane, which will render their d_n values zero. To assign coordinate frames to the

robot, we will first look for the joints (as shown). The robot has six degrees of freedom. In this simple robot, all joints are revolute. The first joint (1) is between links 0 (the fixed base) and 1. Joint 2 is between links 1 and 2, etc. First, we will assign z -axes to each joint, followed by x -axes, as discussed before. Please follow the coordinates as shown in Figures 2.27 and 2.28. Figure 2.28 is a line drawing of the robot in Figure 2.27 for simplicity. Notice why the origin of each frame is where it is.

TABLE 2.1 D-H PARAMETERS TABLE

#	θ	d	a	α
1				
2				
3				
4				
5				
6				

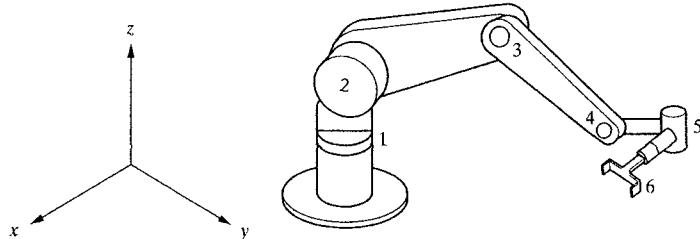


Figure 2.26 A simple articulate robot with six degrees of freedom.

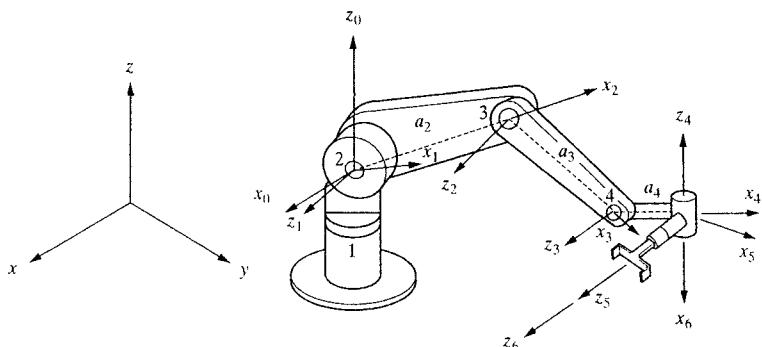


Figure 2.27 Reference frames for the simple six-degree-of-freedom articulate robot.

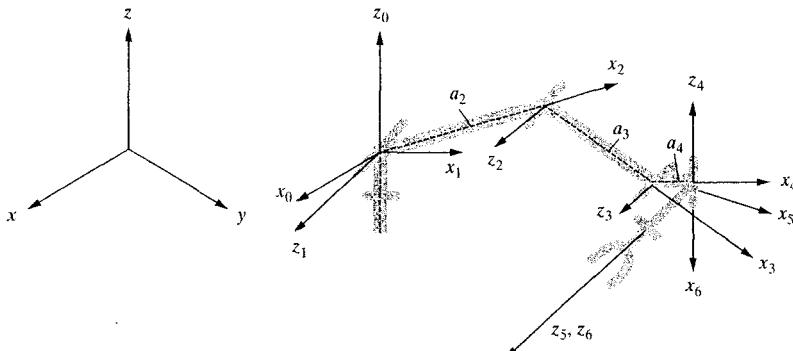


Figure 2.28 Line drawing of the reference frames for the simple six-degree-of-freedom articulate robot.

Start at joint 1. z_0 represents the first joint, which is a revolute joint. x_0 is chosen to be parallel to the reference frame x -axis. 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*. However, these two axes do not move. Next, z_1 is assigned at joint 2. x_1 will be normal to z_0 and z_1 , because these two axes are intersecting. x_2 will be 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 represent the motions of 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, because it will allow us to transform out of frame z_5 - x_5 . Also, notice the location of the origins of the first and the last frames. This will determine the total transformation equation of the robot. You may be able to 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. You can verify that this is correct, because, physically, whether the actual joint is a little higher or lower will not make any difference in the robot's movements. Thus, the origin can be as shown without regard to the physical location of the joint on the base.

Next, we will follow the assigned coordinate frames to fill out the parameters in Table 2.2. Please refer to the previous section for the sequence of four movements between any two frames. Starting with z_0 - x_0 , there will be 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 together, and a rotation of α to bring z_0 to z_1 . Notice 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 will be at z_1 - x_1 .

Next, we will rotate about z_1 an angle of θ_2 to bring x_1 to x_2 , and move along the x_2 -axis a distance of a_2 to bring the origins together. Since the z -axes are paral-

TABLE 2.2 PARAMETERS FOR THE ROBOT OF EXAMPLE 2.19

#	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

lel, there is no need to rotate about the x -axis. Please continue with the same logic to the end, and follow the results.

You *must* realize that similar to other machines, robots do not stay in the one configuration shown in a drawing. You need to visualize the motions, even though the schematic is in two dimensions. This means you must realize that the different links and joints of the robot move, as do the reference frames attached to them. If in this instant, due to the configuration in which the robot is drawn, the axes are shown to be in a particular position and orientation, they will be at other points and orientations as the robot moves. 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 one determines the parameters.

θ represents the joint variable for revolute joints, and d represents the joint variable for prismatic joints. Since this is an all-revolute robot, all joint variables are angles.

The transformations between each two successive joints can be written by simply substituting the parameters from the parameters table into the A matrix. For example, A_1 between frames 0 and 1 (with $\sin 90 = 1$ and $\cos 90 = 0$ for $\alpha = 90^\circ$ and designating C_1 as $\cos \theta_1$, etc.), as well as A_2 through A_6 for the other joints is

$$\begin{aligned}
 A_1 &= \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, & A_2 &= \begin{bmatrix} C_2 & -S_2 & 0 & C_2 a_2 \\ S_2 & C_2 & 0 & S_2 a_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
 A_3 &= \begin{bmatrix} C_3 & -S_3 & 0 & C_3 a_3 \\ S_3 & C_3 & 0 & S_3 a_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, & A_4 &= \begin{bmatrix} C_4 & 0 & -S_4 & C_4 a_4 \\ S_4 & 0 & C_4 & S_4 a_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.55)
 \end{aligned}$$

$$A_5 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Please note: To simplify the final solutions, we will use the following trigonometric functions:

$$\begin{aligned} S\theta_1 C\theta_2 + C\theta_1 S\theta_2 &= S(\theta_1 + \theta_2) = S_{12}, \\ C\theta_1 C\theta_2 - S\theta_1 S\theta_2 &= C(\theta_1 + \theta_2) = C_{12}. \end{aligned} \quad (2.56)$$

The total transformation between the base of the robot and the hand is:

$${}^R T_H = A_1 A_2 A_3 A_4 A_5 A_6 \quad (2.57)$$

$$\begin{aligned} & \left[\begin{array}{cccc} C_1(C_{234}C_5C_6 - S_{234}S_6) & C_1(-C_{234}C_5C_6 - S_{234}C_6) & C_1(C_{234}S_5) & C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ -S_1S_5C_6 & +S_1S_5S_6 & +S_1C_5 & \\ \hline S_1(C_{234}C_5C_6 - S_{234}S_6) & S_1(-C_{234}C_5C_6 - S_{234}C_6) & S_1(C_{234}S_5) & S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ +C_1S_5S_6 & -C_1S_5C_6 & -C_1C_5 & \\ \hline S_{234}C_5C_6 + C_{234}S_6 & -S_{234}C_5C_6 + C_{234}C_6 & S_{234}S_5 & S_{234}a_4 + S_{23}a_3 + S_2a_2 \\ 0 & 0 & 0 & 1 \end{array} \right]. \end{aligned}$$

Example 2.20

The Stanford Arm. Assign coordinate frames to the Stanford Arm (Figure 2.29), and fill out the parameters table. The Stanford Arm is a spherical coordinate arm, which

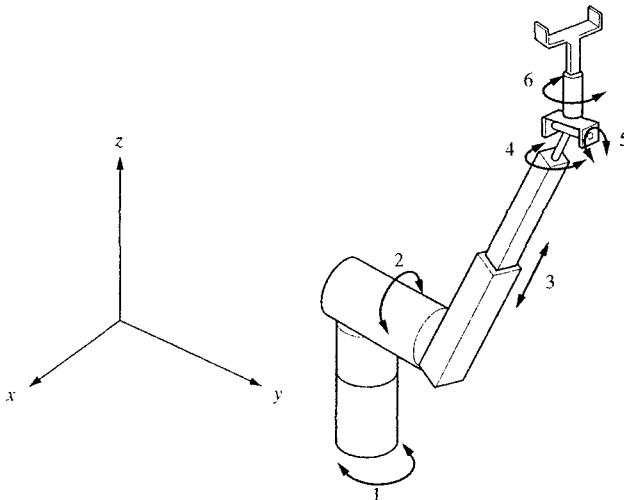


Figure 2.29 Schematic drawing of the Stanford Arm.

means that the first two joints are revolute, while the third is prismatic. The last three wrist joints are all revolute joints.

Solution To allow you to work on this before you see the solution, the solution of 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:

$${}^R T_{H_{\text{STANFORD}}} = {}^0 T_6 = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where

$$\begin{aligned} n_x &= C_1[C_2(C_4C_5C_6 - S_4S_6) - S_2S_5C_6] - S_1(S_4C_5C_6 + C_4S_6), \\ n_y &= S_1[C_2(C_4C_5C_6 - S_4S_6) - S_2S_5C_6] + C_1(S_4C_5C_6 + C_4S_6), \\ n_z &= -S_2(C_4C_5C_6 - S_4S_6) - C_2S_5C_6, \\ o_x &= C_1[-C_2(C_4C_5S_6 + S_4C_6) + S_2S_5S_6] - S_1(-S_4C_5S_6 + C_4C_6), \\ o_y &= S_1[-C_2(C_4C_5S_6 + S_4C_6) + S_2S_5S_6] + C_1(-S_4C_5S_6 + C_4C_6), \\ o_z &= S_2(C_4C_5S_6 + S_4C_6) + C_2S_5S_6, \\ a_x &= C_1(C_2C_4S_5 + S_2C_5) - S_1S_4S_5, \\ a_y &= S_1(C_2C_4S_5 + S_2C_5) + C_1S_4S_5, \\ a_z &= -S_2C_4S_5 + C_2C_5, \\ p_x &= C_1S_2d_3 - S_1d_2, \\ p_y &= S_1S_2d_3 + C_1d_2, \\ p_z &= C_2d_3. \end{aligned} \tag{2.58}$$

2.9 THE INVERSE KINEMATIC SOLUTION OF ROBOTS

As mentioned before, what we are actually interested in is the inverse kinematic solutions. With inverse kinematic solutions, we will be able to determine the value of each joint in order to place the arm at a desired position and orientation. We have already seen the inverse kinematic solutions of specific coordinate systems. In this section, we will see a general procedure for solving the kinematic equations.

As you probably 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 the angles, we will routinely premultiply the ${}^R T_H$ matrix with the individual A_n^{-1} matrices. This will yield the right-hand side of the equation free of the individual angles, allowing us to find elements that yield sines and cosines of angles and, subsequently, the angles.

In this section, a summary of this technique [5] is presented for the simple manipulator arm of Example 2.19. Although the solution presented is for this particular robot with the given configuration, it may similarly be repeated for other robots. As we saw in Example 2.19, the final equation representing the robot is

$${}^R T_H = A_1 A_2 A_3 A_4 A_5 A_6$$

$$= \begin{bmatrix} C_1(C_{234}C_5C_6 - S_{234}S_6) & C_1(-C_{234}C_5C_6 - S_{234}C_6) & C_1(C_{234}S_5) & C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ -S_1S_5C_6 & +S_1S_5S_6 & +S_1C_5 & \\ S_1(C_{234}C_5C_6 - S_{234}S_6) & S_1(-C_{234}C_5C_6 - S_{234}C_6) & S_1(C_{234}S_5) & S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ +C_1S_5C_6 & -C_1S_5S_6 & -C_1C_5 & \\ S_{234}C_5C_6 + C_{234}S_6 & -S_{234}C_5C_6 + C_{234}C_6 & S_{234}S_5 & S_{234}a_4 + S_{23}a_3 + S_2a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

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

$${}^R T_H = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.59)$$

To solve for the angles, we will successively premultiply the two matrices with the A_n^{-1} matrices, starting with A_1^{-1} :

$$A_1^{-1} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_1^{-1}[\text{RHS}] = A_2 A_3 A_4 A_5 A_6, \quad (2.60)$$

$$\begin{bmatrix} C_1 & S_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ S_1 & -C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_2 A_3 A_4 A_5 A_6$$

$$\begin{bmatrix} n_x C_1 + n_y S_1 & o_x C_1 + o_y S_1 & a_x C_1 + a_y S_1 & p_x C_1 + p_y S_1 \\ n_z & o_z & a_z & p_z \\ n_x S_1 - n_y C_1 & o_x S_1 - o_y C_1 & a_x S_1 - a_y C_1 & p_x S_1 - p_y C_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \quad (2.61)$$

$$\begin{bmatrix} C_{234}C_5C_6 - S_{234}S_6 & -C_{234}C_5C_6 - S_{234}C_6 & C_{234}S_5 & C_{234}a_4 + C_{23}a_3 + C_2a_2 \\ S_{234}C_5C_6 + C_{234}S_6 & -S_{234}C_5C_6 + C_{234}C_6 & S_{234}S_5 & S_{234}a_4 + S_{23}a_3 + S_2a_2 \\ -S_5C_6 & S_5S_6 & C_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

From the 3,4 elements of the equation,

$$p_x S_1 - p_y C_1 = 0 \rightarrow \theta_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right) \text{ and } \theta_1 = \theta_1 + 180^\circ. \quad (2.62)$$

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

$$\begin{aligned} p_x C_1 + p_y S_1 &= C_{234} a_4 + C_{23} a_3 + C_2 a_2, \\ p_z &= S_{234} a_4 + S_{23} a_3 + S_2 a_2. \end{aligned} \quad (2.63)$$

Rearranging the two equations, squaring them, and then adding the squares gives

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

Referring to the trigonometric functions of Equation (2.56), we see that

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

Thus,

$$C_3 = \frac{(p_x C_1 + p_y S_1 - C_{234} a_4)^2 + (p_z - S_{234} a_4)^2 - a_2^2 - a_3^2}{2a_2 a_3}. \quad (2.64)$$

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

$$\theta_3 = \tan^{-1} \frac{S_3}{C_3}. \quad (2.65)$$

Since joints 2,3, and 4 are parallel, premultiplying by inverses of A_2 and A_3 will not yield useful results. The next step is to premultiply by the inverses of A_1 through A_4 , which results in:

$$A_4^{-1} A_3^{-1} A_2^{-1} A_1^{-1} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_4^{-1} A_3^{-1} A_2^{-1} A_1^{-1} [\text{RHS}] = A_5 A_6, \quad (2.66)$$

which, in turn, yields

$$\begin{aligned} \begin{bmatrix} C_{234}(C_1 n_x + S_1 n_y) & C_{234}(C_1 o_x + S_1 o_y) & C_{234}(C_1 a_x + S_1 a_y) & C_{234}(C_1 p_x + S_1 p_y) + \\ + S_{234} n_z & + S_{234} o_z & + S_{234} a_x & S_{234} p_z - C_{34} a_2 - C_4 a_3 - a_4 \\ C_1 n_y - S_1 n_x & C_1 o_y - S_1 o_x & C_1 a_y - S_1 a_x & 0 \\ -S_{234}(C_1 n_x + S_1 n_y) & -S_{234}(C_1 o_x + S_1 o_y) & -S_{234}(C_1 a_x + S_1 a_y) & -S_{234}(C_1 p_x + S_1 p_y) + \\ + C_{234} n_z & + C_{234} o_z & + C_{234} a_z & C_{234} p_z + S_{34} a_2 + S_4 a_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ = \begin{bmatrix} C_5 C_6 & -C_5 S_6 & S_5 & 0 \\ S_5 C_6 & -S_5 S_6 & -C_5 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \quad (2.67)$$

From the 3,3 elements of the matrices in Equation (2.67),

$$\begin{aligned} -S_{234}(C_1a_x + S_1a_y) + C_{234}a_z &= 0 \quad \rightarrow \\ \theta_{234} = \tan^{-1}\left(\frac{a_z}{C_1a_x + S_1a_y}\right) \quad \text{and} \quad \theta_{234} &= \theta_{234} + 180^\circ, \end{aligned} \quad (2.68)$$

and we can calculate S_{234} and C_{234} , which are used to calculate θ_3 , as previously discussed.

Now, referring again to Equation (2.63), 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

$$\begin{cases} p_xC_1 + p_yS_1 - C_{234}a_4 = (C_2C_3 - S_2S_3)a_3 + C_2a_2, \\ p_z - S_{234}a_4 = (S_2C_3 + C_2S_3)a_3 + S_2a_2. \end{cases} \quad (2.69)$$

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

$$\begin{cases} S_2 = \frac{(C_3a_3 + a_2)(p_z - S_{234}a_4) - S_3a_3(p_xC_1 + p_yS_1 - C_{234}a_4)}{(C_3a_3 + a_2)^2 + S_3^2a_3^2}, \\ C_2 = \frac{(C_3a_3 + a_2)(p_xC_1 + p_yS_1 - C_{234}a_4) + S_3a_3(p_z - S_{234}a_4)}{(C_3a_3 + a_2)^2 + S_3^2a_3^2}. \end{cases} \quad (2.70)$$

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

$$\theta_2 = \tan^{-1} \frac{(C_3a_3 + a_2)(p_z - S_{234}a_4) - S_3a_3(p_xC_1 + p_yS_1 - C_{234}a_4)}{(C_3a_3 + a_2)(p_xC_1 + p_yS_1 - C_{234}a_4) + S_3a_3(p_z - S_{234}a_4)}. \quad (2.71)$$

Now that θ_2 and θ_3 are known, we can calculate

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3. \quad (2.72)$$

Please remember that since there are two solutions for θ_{234} (Equation 2.68), there will be two solutions for θ_4 as well.

From 1,3 and 2,3 elements of Equation (2.67), we get

$$\begin{cases} S_5 = C_{234}(C_1a_x + S_1a_y) + S_{234}a_z, \\ C_5 = -C_1a_y + S_1a_x, \end{cases} \quad (2.73)$$

and $\theta_5 = \tan^{-1} \frac{(C_{234}(C_1a_x + S_1a_y) + S_{234}a_z)}{S_1a_x - C_1a_y}. \quad (2.74)$

As you have probably noticed, there is no decoupled equation for θ_6 . As a result, we will have to premultiply Equation (2.67) by the inverse of A_5 to decouple it. Doing so, we get

$$\begin{bmatrix} C_5[C_{234}(C_1n_x + S_1n_y) + S_{234}n_z] & C_5[C_{234}(C_1o_x + S_1o_y) + S_{234}o_z] & 0 & 0 \\ -S_5(S_1n_x - C_1n_y) & -S_5(S_1o_x - C_1o_y) & & \\ -S_{234}(C_1n_x + S_1n_y) + C_{234}n_z & -S_{234}(C_1o_x + S_1o_y) + C_{234}o_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.75)$$

From the 2,1 and 2,2 elements of Equation (2.75), we get

$$\theta_6 = \tan^{-1} \frac{-S_{234}(C_1n_x + S_1n_y) + C_{234}n_z}{-S_{234}(C_1o_x + S_1o_y) + C_{234}o_z}. \quad (2.76)$$

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

It is important to notice 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, and, as a result, one will have to solve the matrices directly or by calculating the inverse of the matrix and solving for the unknowns. Most industrial robots have intersecting wrist joints.

2.10 INVERSE KINEMATIC PROGRAMMING OF ROBOTS

The equations we found for solving the inverse kinematic problem of robots can directly be used to drive the robot to a position. In fact, no robot would actually use the forward kinematic equations to solve for these results. The only equations that are used are the set of six (or fewer, depending on the number of joints) equations that calculate the joint values. In other words, the robot designer must calculate the inverse solution and derive these equations and, in return, use them to drive the robot to position. This is necessary for the practical reason that it takes a long time for a computer to calculate the inverse of the forward kinematic equations or to substitute values into them and calculate the unknowns (joint variables) by methods such as Gaussian elimination.

For a robot to move in a predictable path, say, a straight line, it is necessary to recalculate joint variables many times a second. Imagine that a robot needs to move in the straight line between a starting point *A* and a destination point *B*. If no other action is taken and the robot moves from point *A* to point *B*, the path is unpredictable. The robot moves all its joints until they are all at the final value, which will place the robot at the destination point *B*. However, depending on the rate of change in each joint, the hand will follow an unknown path in between the two

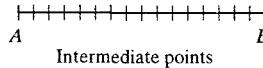


Figure 2.30 Small sections of movement for straight-line motions.

points. To make the robot follow a straight line, it is necessary to break the line into many small sections (Figure 2.30) and make the robot follow those very small sections sequentially between the two points. This means that a new solution must be calculated for each small section. Typically, the location may be recalculated between 50 to 200 times a second. This means that if calculating a solution takes more than 5 to 20 ms [10], the robot will lose accuracy or will not follow the specified path. The shorter the time it takes to calculate a new solution, the more accurate the robot can be. As a result, it is vital to eliminate all unnecessary computations, as much as possible, to allow the computer controller to calculate more solutions. This is why the designer must do all mathematical manipulations beforehand, and only program the robot controller to calculate the final solutions. This will be discussed in more detail in Chapter 5.

In the case of the revolute 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 the controller needs to use to calculate the unknown angles is the set of inverse solutions as follows:

$$\begin{aligned} \theta_1 &= \tan^{-1}\left(\frac{p_y}{p_x}\right) \quad \text{and} \quad \theta_1 = \theta_1 + 180^\circ, \\ \theta_{234} &= \tan^{-1}\left(\frac{a_z}{C_1 a_x + S_1 a_y}\right) \quad \text{and} \quad \theta_{234} = \theta_{234} + 180^\circ, \\ C_3 &= \frac{(p_x C_1 + p_y S_1 - C_{234} a_4)^2 + (p_z - S_{234} a_4)^2 - a_2^2 - a_3^2}{2 a_2 a_3}, \\ S_3 &= \pm \sqrt{1 - C_3^2}, \\ \theta_3 &= \tan^{-1} \frac{S_3}{C_3}, \\ \theta_2 &= \tan^{-1} \frac{(C_3 a_3 + a_2)(p_z - S_{234} a_4) - S_3 a_3 (p_x C_1 + p_y S_1 - C_{234} a_4)}{(C_3 a_3 + a_2)(p_x C_1 + p_y S_1 - C_{234} a_4) + S_3 a_3 (p_z - S_{234} a_4)}, \\ \theta_4 &= \theta_{234} - \theta_2 - \theta_3, \\ \theta_5 &= \tan^{-1} \frac{C_{234}(C_1 a_x + S_1 a_y) + S_{234} a_z}{S_1 a_x - C_1 a_y}, \\ \theta_6 &= \tan^{-1} \frac{-S_{234}(C_1 n_x + S_1 n_y) + C_{234} n_z}{-S_{234}(C_1 o_x + S_1 o_y) + C_{234} o_z}. \end{aligned} \tag{2.77}$$

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. You notice that all operations in this computation are simple arithmetic or trigonometric operations.

2.11 DEGENERACY AND DEXTERITY

Degeneracy. Degeneracy occurs when the robot loses a degree of freedom and thus cannot perform as desired [11]. This occurs under two conditions: (1) when the robot's joints reach their physical limits, and as a result, cannot move any further, and (2) a robot may become degenerate in the middle of its workspace if the z-axes of two similar joints become colinear. This means that at this instant, whichever joint moves, the same motion will result. As a result, the controller will not know which joint to move. Since in either case the total number of degrees of freedom available is fewer than six, there is no solution for the robot. In the case of colinear joints, the determinant of the position matrix is zero as well. Figure 2.31 shows a simple robot in a vertical configuration, where joints 1 and 6 are collinear. As you can see, whether 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. Please notice that this condition occurs if the two joints are similar. Otherwise, if one joint is prismatic and one is revolute (as in

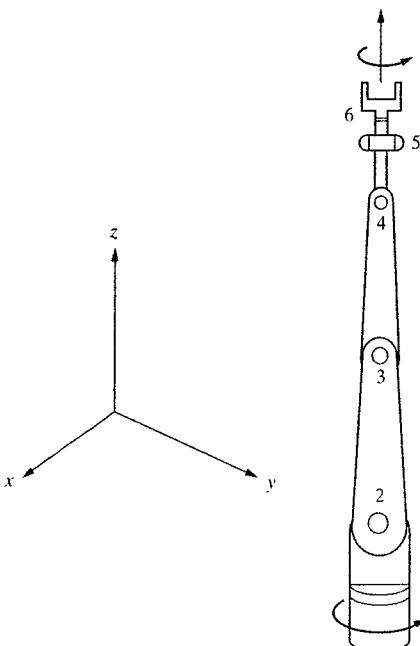


Figure 2.31 An example of a robot in a degenerate position.

joints 3 and 4 of the Stanford arm), although the z -axes are colinear, the robot will not be in degenerate condition.

Paul [11] has shown that if $\sin(\alpha_4)$, $\sin(\alpha_5)$ or $\sin(\theta_5)$ are zero, the robot will be degenerate. Obviously, α_4 and α_5 can be designed such as to prevent the degeneracy of the robot. However, anytime θ_5 approaches zero or 180° , the robot will become degenerate.

Dexterity. One is supposed to be able to position and orientate a six-degree-of-freedom 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, it will get to a point where although it is possible to locate it at a desired point, it will be impossible to orientate it at desired orientations. The volume of points where one can position the robot as desired, but not orientate it is called nondexterous volume.

2.12 THE FUNDAMENTAL PROBLEM WITH D–H REPRESENTATION

Although D–H representation has been extensively used in modeling and analysis of robot motions and although it has become a standard method for doing so, 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. Thus, if there is any motion about the y -axis, the method will fail. This can occur in a number of circumstances. For example, suppose that two joint axes which are supposed to be parallel are assembled with a slight deviation. The small angle between the two axes will require a motion about the y -axis. Since all industrial real robots have some degree of inaccuracy in their manufacture, their inaccuracy cannot be modeled with D–H.

Example 2.20 (continued)

Reference Frames for Stanford Arm: Figure 2.32 is the solution for the Stanford Arm of Example 2.20 (Figure 2.29). It is simplified for improved visibility.

For the derivation of the inverse kinematic solution of Stanford Arm, please refer to References [5,13]. The following is a summary of the inverse kinematic solution for Stanford Arm:

$$\theta_1 = \tan^{-1} \left(\frac{p_y}{p_x} \right) - \tan^{-1} \frac{d_2}{\pm \sqrt{r^2 - d_2^2}} \quad \text{where} \quad r = \sqrt{p_x^2 + p_y^2}, \quad (2.78)$$

$$\theta_2 = \tan^{-1} \frac{C_1 p_x + S_1 p_y}{p_z}, \quad (2.79)$$

$$d_3 = S_2(C_1 p_x + S_1 p_y) + C_2 p_z, \quad (2.80)$$

$$\theta_4 = \tan^{-1} \frac{-S_1 a_x + C_1 a_y}{C_2(C_1 a_x + S_1 a_y) - S_2 a_z} \quad \text{and} \quad \theta_4 = \theta_4 + 180^\circ \quad \text{if} \quad \theta_5 < 0, \quad (2.81)$$

$$\theta_5 = \tan^{-1} \frac{C_4[C_2(C_1a_x + S_1a_y) - S_2a_z] + S_4[-S_1a_x + C_1a_y]}{S_2(C_1a_x + S_1a_y) + C_2a_z}, \quad (2.82)$$

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

$$\begin{aligned} S_6 &= -C_5\{C_4[C_2(C_1o_x + S_1o_y) - S_2o_z] + S_4[-S_1o_x + C_1o_y]\} \\ &\quad + S_5\{S_2(C_1o_x + S_1o_y) + C_2o_z\}, \\ C_6 &= -S_4[C_2(C_1o_x + S_1o_y) - S_2o_z] + C_4[-S_1o_x + C_1o_y]. \end{aligned} \quad (2.83)$$

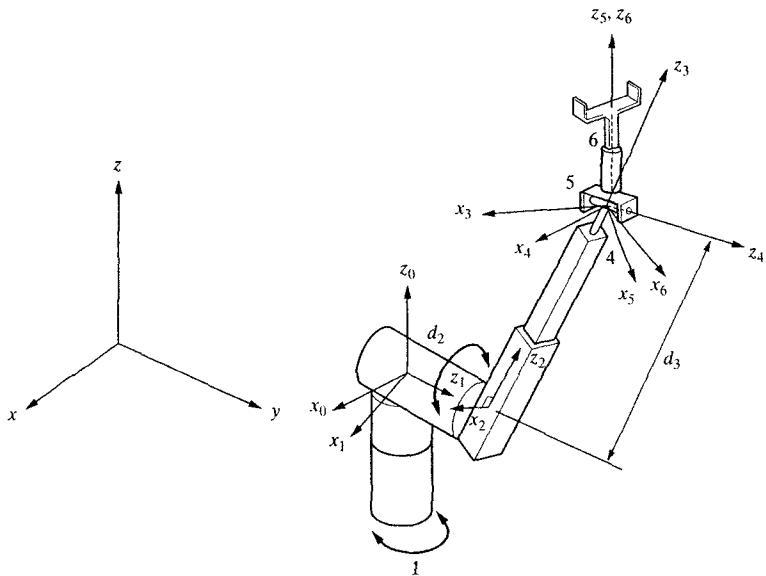


Figure 2.32 The frames of the Stanford Arm.

TABLE 2.3 THE PARAMETERS TABLE FOR THE STANFORD ARM

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

2.13 DESIGN PROJECT 1: A THREE-DEGREE-OF-FREEDOM ROBOT

Starting with this chapter and continuing with the rest of the book, we will apply the current information in each chapter to the design of a simple robot. This will help you apply the current material to the design of a robot of your own. Since common six-degree-of-freedom robots are too complicated to be a simple robot, we will consider a three-degree-of-freedom one. 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 a robot and its configuration, keeping in mind the possible types of actuators you may want to consider later. Although we will 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 (e.g., wood dowels, hollow aluminum, or brass tubes available in hardware stores).

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 your own.. After the configuration of the robot is finalized, you should proceed with the derivation of the forward and inverse kinematic equations. The final result of this part of the design project will be a set of inverse kinematic equations for the simple three-degree-of-freedom robot that can later be used to drive the robot to desired position. You must realize that the price we pay for this simplicity is that we may only specify the position of the robot, but not its orientation.

One of the important considerations in the design of the robot is its joints. Figure 2.33(a) 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 2.33 (b). The latter also allows larger range of motion. On the other hand, although we apparently have to deal with a joint offset d with the joint design in Figure 2.33(b), 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 will assume that the joints of our robot can be built as in Figure 2.33(b) without having to worry about joint offset d .

We will discuss actuators in Chapter 6. However, for this design project, you should probably consider the use of a servomotor or a stepper motor. While you are

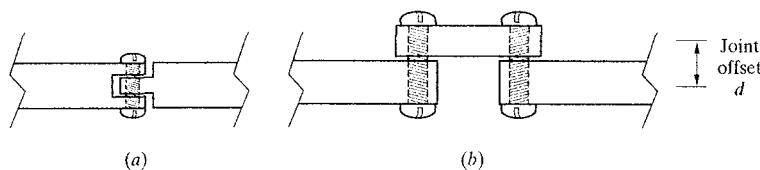


Figure 2.33 Two simple designs for a joint.

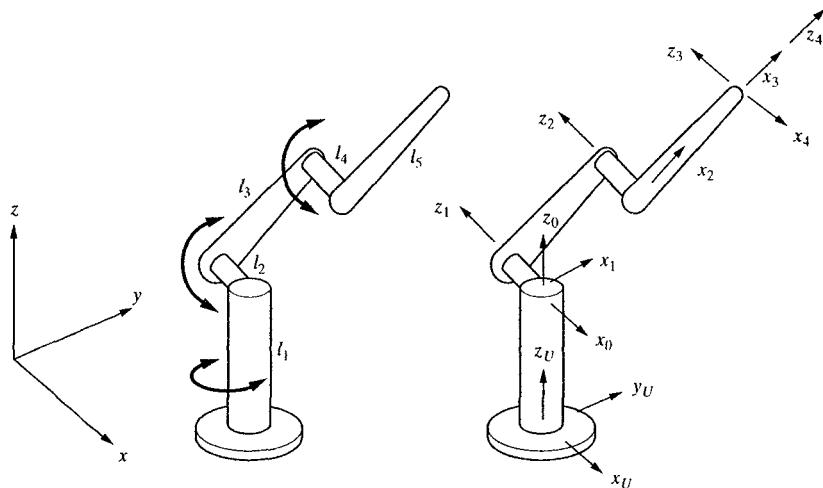


Figure 2.34 A simple three-degree-of-freedom robot design that may be used for the design project.

designing your robot, please do consider what type of actuators you will use and how you will connect the actuators to the links and joints. Still, please remember that at this point, you are only designing the robot configuration and that you can always change your actuators and adapt the new design to 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 frame transformation, and calculate the final ${}^U T_H$. Then using the methods learned in this chapter, develop the inverse kinematic equations of the robot. This means that using these equations, if you actually build the robot, you will be able to run the robot and control its position. (Since the robot has three degrees of freedom, you will not be able to control its orientation.)

Figure 2.34 shows a simple design for a three-degree-of-freedom robot that you may use as a guide for your design. In one student design, the lengths were 8, 2, 9, 2, and 9 inches, respectively. The links were made of hollow aluminum bars, actuated by three DC gearmotors with encoder feedback, connected to the joint through worm gears.

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

2.14 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 mo-

tions of a multiple-degree-of-freedom robot in space and how to derive the forward and inverse kinematic equations of the robot using Denavit–Hartenberg representation technique. This method can be used to represent any type of robot configuration, regardless of number or type of joints or joint and link offset, or twist.

In the next chapter, we will continue with the differential motions of robots, which, in effect, is 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., and Saeed B. Niku, "Robot Arm Positional Deflection Control with a Laser Light," *Proceedings of the Mechatronics '98 Conference*, Skovde, Sweden, Adolfsson and Karlsson, 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*, 2d Edition, Addison Wesley, 1989.
7. Shahinpoor, Mohsen, *A Robot Engineering Textbook*, Harper & Row, Publishers, 1987.
8. Koren, Yoram, *Robotics for Engineers*, McGraw-Hill, 1985.
9. Fu, K. S., R.C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill, 1987.
10. Eman, Kornel F., "Trajectories," *International Encyclopedia of Robotics: Applications and Automation*, Richard C. Dorf, Editor, John Wiley & Sons, New York, 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, and W. A. Watson, "Micras, Microcomputer Interactive Codes for Robot Analysis and Simulation," *Mechanisms and Machine Theory*, Vol. 20, No. 4, 1985, pp. 271–284.

The following Isometric grid is provided to you for use with problems in this chapter. It is meant to be used as a tracing grid for drawing three-dimensional shapes and objects such as robots, frames, and transformations. Please make copies of the grid for each problem that requires graphical representation of the results. The grid is also available commercially.

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

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

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

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

- 17.** A frame ${}^U B$ was moved along its own n -axis a distance of 5 units and then rotated about its o -axis an angle of 60° , followed by a rotation of about the z -axis; it was then translated about its a -axis 3 units and finally rotated about x -axis 45° .

- (a) Calculate the total transformation performed.
- (b) What angles and movements would we have to make if we were to create the same location and orientation using Cartesian and RPY configurations?

- 18.** Frames describing the base of a robot and an object are given relative to the Universe frame.

- (a) Find a transformation ${}^R T_H$ of the robot configuration if the hand of the robot is to be placed on the object.
- (b) By inspection, show whether this robot can be a three-axis spherical robot, and, if so, find α, β, r .
- (c) Assuming that the robot is a six-axis robot with Cartesian and Euler coordinates, find $x, y, z, \phi, \theta, \psi$.

$${}^U T_{\text{obj}} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^U T_R = \begin{bmatrix} 0 & -1 & 0 & 2 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 19.** A robot arm with three degrees of freedom has been designed for applying paint on flat walls as shown in Figure P.2.19.

- (a) Assign coordinate frames as necessary based on D-H representation.
- (b) Fill out the parameters table.
- (c) Find the ${}^U T_H$ matrix.

- 20.** The robot shown in Figure P.2.20 has two degrees of freedom, and the transformation matrix ${}^H T_H$ 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.

- (a) Derive the inverse kinematic equations for θ_1 and θ_2 in symbolic form.
- (b) Calculate the values of θ_1 and θ_2 for the given location.

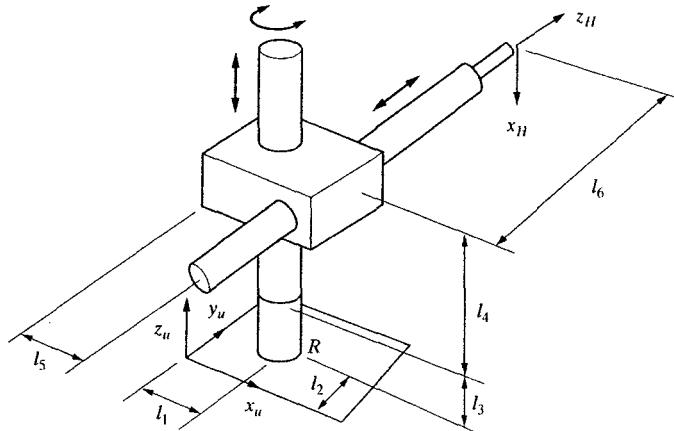


Figure P.2.19

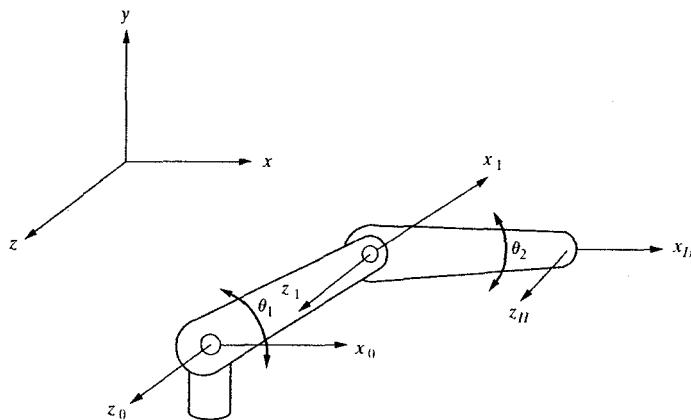
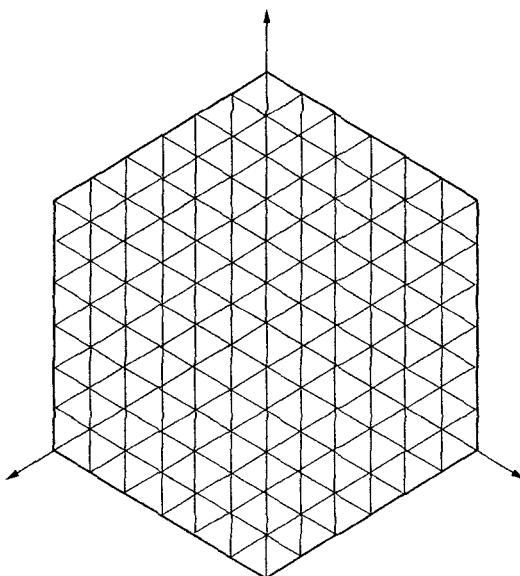


Figure P.2.20

$${}^0T_H = \begin{bmatrix} C_{12} & -S_{12} & 0 & l_2C_{12} + l_1C_1 \\ S_{12} & C_{12} & 0 & l_2S_{12} + l_1S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.2924 & -0.9563 & 0 & 0.6978 \\ 0.9563 & -0.2924 & 0 & 0.8172 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

21. For the SCARA-type robot shown in Figure P.2.21.
- Assign the coordinate frames based on D-H representation.
 - Fill out the parameters table.
 - Write all the \$A\$ matrices.
 - Write the \${}^0T_H\$ matrix in terms of the \$A\$ matrices.



PROBLEMS

- Suppose that instead of a frame, a point $P = (3,5,7)^T$ in space was translated a distance of $d = (2,3,4)^T$. Find the new location of the point relative to the reference frame.
- The following was moved a distance of $d = (5,2,6)^T$:

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

Find the new location of the frame relative to the reference frame.

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

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

- Derive the matrix that represents a pure rotation about the y -axis of the reference frame.
- Derive the matrix that represents a pure rotation about the z -axis of the reference frame.

6. Verify that the rotation matrices about the reference frame axes follow the required constraint equations set by orthogonality and length requirements of directional unit vectors.
7. Find the coordinates of point $P(2,3,4)^T$ relative to the reference frame after a rotation of 45° about the x -axis.
8. Find the coordinates of point $P(3,5,7)^T$ relative to the reference frame after a rotation of 30° about the z -axis.
9. A point P in space is defined as ${}^B P = [5,3,4]^T$ relative to frame B and 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 three-dimensional grid provided in this chapter, plot the transformations and the result and verify it. Also verify graphically that you would not get the same results if you apply 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.
10. 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 three-dimensional grid, plot the transformations and the result and verify it:
- Rotate 90° about the x -axis.
 - Then, rotate 90° about the local a -axis.
 - Then translate 3 units about the y -axis, 6 units about the z -axis, and 5 units about the x -axis.
11. Show that rotation matrices about the y -axis and the z -axis are unitary.
12. Calculate the inverse of the following transformation matrices:

$$T_1 = \begin{bmatrix} 0.527 & -0.574 & 0.628 & 2 \\ 0.369 & 0.819 & 0.439 & 5 \\ -0.766 & 0 & 0.643 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_2 = \begin{bmatrix} 0.92 & 0 & 0.39 & 5 \\ 0 & 1 & 0 & 6 \\ -0.39 & 0 & 0.92 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

13. Write the correct sequence of movements that must be made in order to “unrotate” the spherical coordinates and to make it parallel to the reference frame. About what axes are these rotations supposed to be?
14. A spherical coordinate system is used to describe the position of the hand of a robot. In a certain situation, the hand is later “unrotated” back to be parallel to the reference frame, and the matrix representing it is described as

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

- (a) Find the necessary values of $\bar{n}, \bar{\beta}, \bar{\gamma}$ to achieve this location.
- (b) Find the components of the original matrix $\bar{n}, \bar{\alpha}, \bar{a}$ vectors for the hand before it was unrotated.

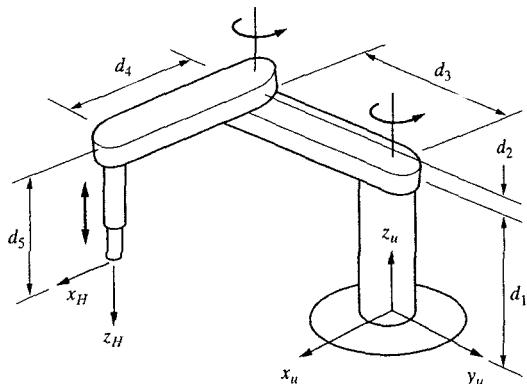


Figure P.2.21

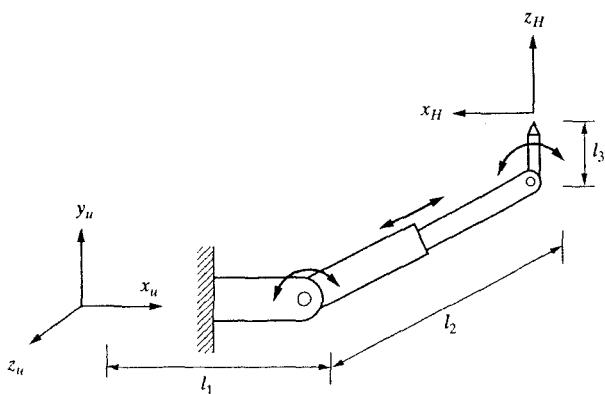


Figure P.2.22

22. A special three-degree-of-freedom spraying robot has been designed as shown in Figure P.2.22.
 - (a) Assign the coordinate frames based on D-H representation.
 - (b) Fill out the parameters table.
 - (c) Write all the A matrices.
 - (d) Write the 0T_H matrix in terms of the A matrices.
23. For the Unimation Puma 562, six-axis robot shown in Figure P.2.23,
 - (a) Assign the coordinate frames based on D-H representation.
 - (b) Fill out the parameters table.

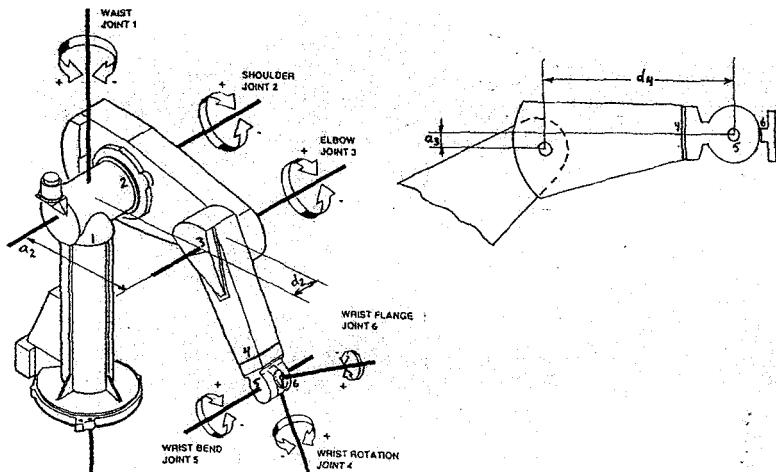


Figure P.2.23 Picture of Puma 562 is reprinted with permission from Staubli Robotics.

#	θ	d	a	α
1				
2				
3				
4				
5				
6				

(c) Write all the A matrices.

(d) Find the ${}^R T_H$ matrix for the following values:

$$\text{Base height} = 27 \text{ in}, d_1 = 6 \text{ in}, a_2 = 15 \text{ in}, a_3 = 1 \text{ in}, d_4 = 18 \text{ in}, d_6 = 5 \text{ in},$$

$$\theta_1 = 0^\circ, \theta_2 = 45^\circ, \theta_3 = 0^\circ, \theta_4 = 0^\circ, \theta_5 = -45^\circ, \theta_6 = 0^\circ$$

24. For the four-degree-of-freedom robot depicted in Figure P.2.24,

(a) Assign appropriate frames for D-H representation.

(b) Fill out the parameters table.

(c) Write an equation in terms of A matrices that shows how ${}^U T_H$ can be calculated.

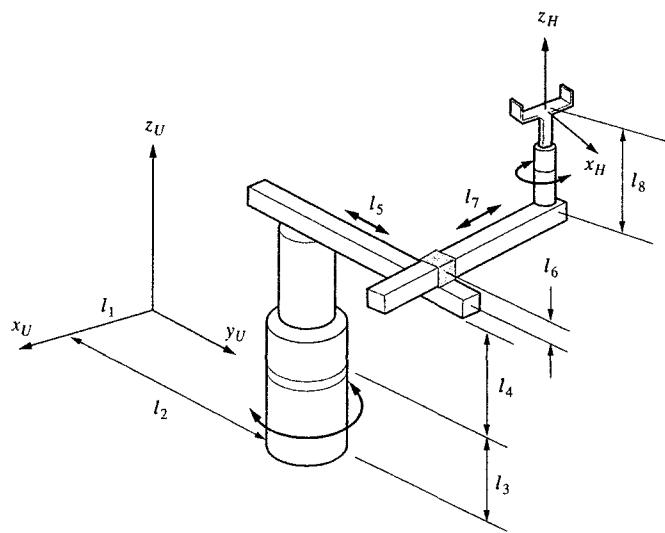


Figure P.2.24

#	θ	d	a	α
1				
2				
3				
4				

3

Differential Motions and Velocities

3.1 INTRODUCTION

Differential motions are small movements of mechanisms (in this case the robots) that can be used to derive velocity relationships between different parts of the mechanism. A differential motion is, by definition, a small movement. Thus, if it is measured in (or calculated for) a small period of time (a differential, or small, time), a velocity relationship can be found.

In this chapter, we will 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. This chapter does contain a fair amount of velocity terms that you should have seen in your dynamics course. If you do not remember the material very well, a review of it is recommended before you continue.

3.2 DIFFERENTIAL RELATIONSHIPS

First, let's see what the differential relationships are. To do this, we will consider a simple mechanism with two degrees of freedom, as shown in Figure 3.1, where each link can independently rotate. 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. This would be similar to a robot, where each link's movement is measured relative to a current frame attached to the previous link.

The velocity of point B can be calculated as follows:

$$\begin{aligned}\bar{V}_B &= \bar{V}_A + \bar{V}_{B/A} \\ &= l_1\dot{\theta}_1[\perp \text{ to } l_1] + l_2(\dot{\theta}_1 + \dot{\theta}_2)[\perp \text{ to } l_2] \\ &= -l_1\dot{\theta}_1 \sin \theta_1 \hat{i} + l_1\dot{\theta}_1 \cos \theta_1 \hat{j} - l_2(\dot{\theta}_1 + \dot{\theta}_2) \\ &\quad \times \sin(\theta_1 + \theta_2) \hat{i} + l_2(\dot{\theta}_1 + \dot{\theta}_2) \cos(\theta_1 + \theta_2) \hat{j}.\end{aligned}\tag{3.1}$$

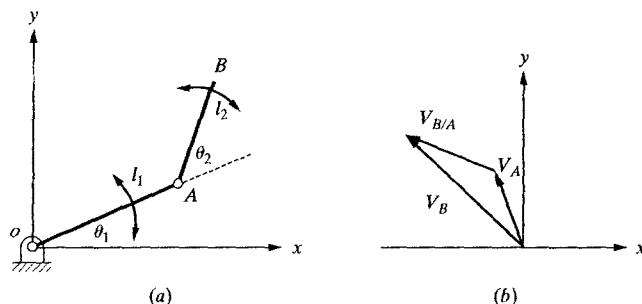


Figure 3.1 (a) A two-degree-of-freedom planar mechanism and (b) a Velocity diagram.

Writing the velocity equation in a matrix form yields the following:

$$\begin{bmatrix} \bar{V}_{B_x} \\ \bar{V}_{B_y} \end{bmatrix} = \begin{bmatrix} -l_1 \sin \theta_1 - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}. \quad (3.2)$$

The left side of the equation represents the \$x\$ and \$y\$ components of the velocity of point \$B\$. As you can see, if the elements of the right-hand-side matrix are multiplied by the corresponding angular velocities of the two links, the velocity of point \$B\$ can be found.

Next, instead of deriving the velocity equation directly from the velocity relationship, we will try to find the same velocity relationship by differentiating the equation that describes the position of point \$B\$, as follows:

$$\begin{cases} x_B = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ y_B = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{cases} \quad (3.3)$$

Differentiating this equation with respect to the two variables \$\theta_1\$ and \$\theta_2\$ yields

$$\begin{cases} dx_B = -l_1 \sin \theta_1 d\theta_1 - l_2 \sin(\theta_1 + \theta_2)(d\theta_1 + d\theta_2), \\ dy_B = l_1 \cos \theta_1 d\theta_1 + l_2 \cos(\theta_1 + \theta_2)(d\theta_1 + d\theta_2), \end{cases} \quad (3.4)$$

and, in matrix form,

$$\begin{bmatrix} dx_B \\ dy_B \end{bmatrix} = \begin{bmatrix} -l_1 \sin \theta_1 - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \end{bmatrix}. \quad (3.5)$$

Differential motion of B

Jacobian

Differential motion of joints

Please notice the similarities between Equations (3.2) and (3.5). You notice that the two equations are similar in content and in form. The difference is that Equa-

tion (3.2) is the velocity relationship, whereas Equation (3.5) is the differential motion relationship. If both sides of Equation (3.5) are divided by dt , since dx_B/dt is V_{Bx} and $d\theta_1/dt$ is $\dot{\theta}_1$, etc., Equation (3.6) will be exactly the same as Equation (3.2), as follows:

$$\begin{bmatrix} dx_B \\ dy_B \end{bmatrix} \Bigg/ dt = \begin{bmatrix} -l_1 \sin \theta_1 - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \end{bmatrix} \Bigg/ dt. \quad (3.6)$$

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 by the same technique.

3.3 JACOBIAN

The Jacobian is a representation of the geometry of the elements of a mechanism in time. It allows the conversion of differential motions or velocities of individual joints to differential motions or velocities of points of interest. It also relates the individual joint motions to overall mechanism motions. Jacobian is time related; since the values of θ_1 and θ_2 vary in time, the magnitude of the elements of the Jacobian vary in time as well.

As you noticed in Section 3.2, the Jacobian was formed from the elements of the position equations that were differentiated with respect to θ_1 and θ_2 . Thus, the Jacobian can be calculated by taking the derivatives of each position equation with respect to all variables.

Suppose that we have a set of equations Y_i in terms of a set of variables x_j :

$$Y_i = f_i(x_1, x_2, x_3, \dots, x_j). \quad (3.7)$$

The differential change in Y_i for a differential change in x_j is

$$\left\{ \begin{array}{l} \delta Y_1 = \frac{\partial f_1}{\partial x_1} \delta x_1 + \frac{\partial f_1}{\partial x_2} \delta x_2 + \dots + \frac{\partial f_1}{\partial x_j} \delta x_j, \\ \delta Y_2 = \frac{\partial f_2}{\partial x_1} \delta x_1 + \frac{\partial f_2}{\partial x_2} \delta x_2 + \dots + \frac{\partial f_2}{\partial x_j} \delta x_j, \\ \vdots \\ \delta Y_i = \frac{\partial f_i}{\partial x_1} \delta x_1 + \frac{\partial f_i}{\partial x_2} \delta x_2 + \dots + \frac{\partial f_i}{\partial x_j} \delta x_j. \end{array} \right. \quad (3.8)$$

Equation (3.8) can be written in a matrix form, and it represents the differential relationship between individual variables and the functions. The matrix encompassing this relationship is called Jacobian, as shown in Equation (3.9). Thus, the Jacobian can be calculated by taking the derivative of each equation with respect to all variables. We will apply the same principle for the calculation of the Jacobian of a robot.

$$\begin{bmatrix} \delta Y_1 \\ \delta Y_2 \\ \vdots \\ \delta Y_i \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_j} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \cdots & \vdots \\ \vdots & & & \vdots \\ \frac{\partial f_i}{\partial x_1} & & \frac{\partial f_i}{\partial x_j} & \end{bmatrix} \begin{bmatrix} \delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_j \end{bmatrix}, \text{ or } [\delta Y_i] = \left[\frac{\partial f_i}{\partial x_j} \right] [\delta x_j] \quad (3.9)$$

Using the same relationship as before and differentiating the position equations of a robot, we can write the following equation, which relates joint differential motions of a robot to the differential motion of its hand frame:

$$\begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \\ \delta z \end{bmatrix} = \begin{bmatrix} & & & & d\theta_1 \\ & & & & d\theta_2 \\ Robot & Jacobian & & & d\theta_3 \\ & & & & d\theta_4 \\ & & & & d\theta_5 \\ & & & & d\theta_6 \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \\ d\theta_3 \\ d\theta_4 \\ d\theta_5 \\ d\theta_6 \end{bmatrix}, \text{ or } [D] = [J][D_\theta], \quad (3.10)$$

where dx , dy , and dz in $[D]$ represent the differential motions of the hand along the x -, y -, and z -axes, respectively, δx , δy , and δz in $[D]$ represent the differential rotations of the hand around the x -, y -, and z -axes, respectively, and $[D_\theta]$ represents the differential motions of the joints. As was mentioned earlier, if these two matrices are divided by dt , they will represent velocities instead of differential motions. In this chapter, we will work with the differential motions, rather than velocities, since in all relationships, by simply dividing the differential motions by dt , we can get the velocities.

Example 3.1

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:

$$J = \begin{bmatrix} 2 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad D_\theta = \begin{bmatrix} 0 \\ 0.1 \\ -0.1 \\ 0 \\ 0 \\ 0.2 \end{bmatrix}.$$

Solution Substituting the foregoing matrices into Equation (3.10), we get

$$D = JD_\theta = \begin{bmatrix} 2 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0.1 \\ -0.1 \\ 0 \\ 0 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 0 \\ -0.1 \\ 0.1 \\ 0 \\ -0.1 \\ 0.2 \end{bmatrix} = \begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \\ \delta z \end{bmatrix}$$

3.4 DIFFERENTIAL MOTIONS OF A FRAME

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. In the second case, we will look at the differential motions of the mechanism that causes the motions and how it relates to the motions of the frame. As you can see in Figure 3.2(b), the differential motions of the hand frame of the robot are caused by the differential motions in each of the joints of the robot. Thus, as the joints of the robot move a differential amount, the hand frame moves a differential amount as well, moving the frame representing it a differential amount. Thus, we will have to relate the motions of the robot to the motions of the frame.

To understand what this really means, suppose that you have a robot welding two pieces together. For best results, you will want the robot to move at a constant speed. This means that 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 the frame. However, the motion is caused by the robot. (It could actually be caused by something else, but we are using a robot, so we must relate it to the robot's motions.) Thus, 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.

The differential motions of a frame can be divided into the following:

- Differential translations,
- Differential rotations,
- Differential transformations (translations and rotations).

First, we will study these motions.

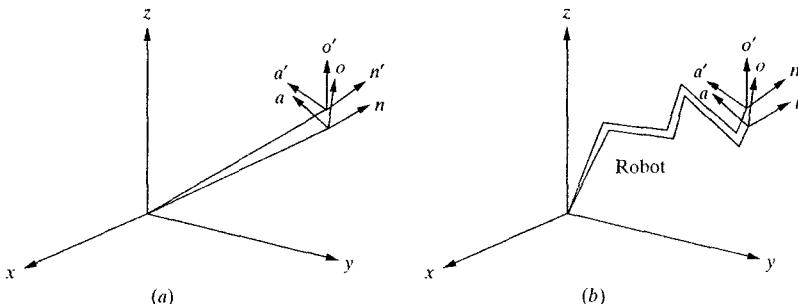


Figure 3.2 (a) Differential motions of a frame and (b) differential motions of a frame as related to the differential motions of a robot.

3.4.1 Differential Translations

A differential translation is a translation of a frame at differential values. Thus, it can be represented by $\text{Trans}(dx, dy, dz)$. This means that the frame has moved a differential amount along the three axes.

3.4.2 Differential Rotations

A differential rotation is a small rotation of the frame. It is generally represented by $\text{Rot}(k, d\theta)$, which means that the frame has rotated an angle of $d\theta$ about an axis \hat{k} .

Specifically, differential rotations about the x -, y -, z -axes are defined by δx , δy , and δz , respectively. Since the rotations are small, we can use the following approximations:

$$\sin \delta x = \delta x \text{ (in radians),}$$

$$\cos \delta x = 1.$$

Then, the rotation matrices representing differential rotations about the x -, y -, and z -axes are

$$\begin{aligned} \text{Rot}(x, \delta x) &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta x & 0 \\ 0 & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, & \text{Rot}(y, \delta y) &= \begin{bmatrix} 1 & 0 & \delta y & 0 \\ 0 & 1 & 0 & 0 \\ -\delta y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ \text{Rot}(z, \delta z) &= \begin{bmatrix} 1 & -\delta z & 0 & 0 \\ \delta z & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \quad (3.11)$$

You will notice that these matrices defy the rule we had established previously about the magnitude of each vector being one unit. 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 to be negligible and are usually neglected. If we do neglect the higher order differentials, such as $(\delta x)^2$, the magnitude of the vectors remain acceptable.

As we have already seen, if the order of multiplication of matrices changes, the result will change as well. Thus, in matrix multiplication, the order of matrices is very important. If we multiply two differential motions in different orders, we will get two different results:

$$\text{Rot}(x, \delta x)\text{Rot}(y, \delta y) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta x & 0 \\ 0 & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \delta y & 0 \\ 0 & 1 & 0 & 0 \\ -\delta y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{aligned} & \begin{bmatrix} 1 & 0 & \delta y & 0 \\ \delta x \delta y & 1 & -\delta x & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ \text{Rot}(y, \delta y) \text{Rot}(x, \delta x) &= \begin{bmatrix} 1 & 0 & \delta y & 0 \\ 0 & 1 & 0 & 0 \\ -\delta y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta x & 0 \\ 0 & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\ & \begin{bmatrix} 1 & \delta x \delta y & \delta y & 0 \\ 0 & 1 & -\delta x & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

However, if, as before, we neglect the higher order differentials, such as $\delta x \delta y$, the results are exactly the same. Thus, in differential motions, we can assume that the order of multiplication is not important and that $\text{Rot}(x, \delta x) \text{Rot}(y, \delta y) = \text{Rot}(y, \delta y) \text{Rot}(x, \delta x)$.

You may remember from your dynamics course that angles cannot be added in different orders, since they are not commutative. However, velocities are, and can be added as, vectors. This is true because, as the preceding section shows, if we neglect the higher order differentials, the order of multiplication will be unimportant. Since velocities are, in fact, differential motions divided by time, the same will be true for velocities.

3.4.3 Differential Rotation about a General Axis \hat{k}

Based on the foregoing discussion, 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 motion about a general axis \hat{k} is composed of three differential motions about the three axes, in any order. Thus, a differential motion about any general axis \hat{k} can be expressed as

$$\begin{aligned} \text{Rot}(k, d\theta) &= \text{Rot}(x, \delta x) \text{Rot}(y, \delta y) \text{Rot}(z, \delta z) \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta x & 0 \\ 0 & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \delta y & 0 \\ 0 & 1 & 0 & 0 \\ -\delta y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta z & 0 & 0 \\ \delta z & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.12) \\ &= \begin{bmatrix} 1 & -\delta z & \delta y & 0 \\ \delta x \delta y + \delta z & -\delta x \delta y \delta z + 1 & -\delta x & 0 \\ -\delta y + \delta x \delta z & \delta x + \delta y \delta z & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

If we neglect all higher order differentials, we get

$$\text{Rot}(k, d\theta) = \text{Rot}(x, \delta x) \text{Rot}(y, \delta y) \text{Rot}(z, \delta z) = \begin{bmatrix} 1 & -\delta z & \delta y & 0 \\ \delta z & 1 & -\delta x & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.13)$$

Example 3.2

Find the total differential transformation caused by small rotations about the three axes of $\delta x = 0.1$, $\delta y = 0.05$, and $\delta z = 0.02$ radians.

Solution Substituting the given rotations in Equation (3.13), we get

$$\text{Rot}(k, \delta\theta) = \begin{bmatrix} 1 & -\delta z & \delta y & 0 \\ \delta z & 1 & -\delta x & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -0.02 & 0.05 & 0 \\ 0.02 & 1 & -0.1 & 0 \\ -0.05 & 0.1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3.4.4 Differential Transformations of a Frame

The differential transformation of a frame is a combination of differential translations and rotations. If we denote the original frame as T and assume that the *change* in the frame T as a result of a differential transformation is expressed as dT , then

$$\begin{aligned} [T + dT] &= [\text{Trans}(dx, dy, dz) \text{Rot}(k, d\theta)] [T], \\ \text{or } [dT] &= [\text{Trans}(dx, dy, dz) \text{Rot}(k, d\theta) - I][T], \end{aligned} \quad (3.14)$$

where I is a unit matrix. $[dT]$ expresses the change in the frame after the differential transformation. Equation (3.14) can be written as

$$\begin{aligned} [dT] &= [\Delta][T], \\ \text{where } [\Delta] &= [\text{Trans}(dx, dy, dz) \times \text{Rot}(k, d\theta) - I]. \end{aligned} \quad (3.15)$$

$[\Delta]$ (or simply Δ) is called *differential operator*. It is the product of differential translations and rotations, minus the unit matrix. Multiplying a frame by the differential operator $[\Delta]$ will result in the change in the frame. The differential operator can be found by multiplying the matrices and subtracting the unit matrix as follows:

$$\Delta = \text{Trans}(dx, dy, dz) \times \text{Rot}(k, d\theta) - I$$

$$= \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta z & \delta y & 0 \\ \delta z & 1 & -\delta x & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.16)$$

$$= \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

As you may notice, 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 yields the changes in a frame.

Example 3.3

Write the differential operator matrix for the following differential transformations:

$$dx = 0.5, dy = 0.3, dz = 0.1 \text{ units, and } \delta x = 0.02, \delta y = 0.04, \delta z = 0.06 \text{ radians.}$$

Solution Substituting the given values into Equation (3.16), we get

$$\Delta = \begin{bmatrix} 0 & -0.06 & 0.04 & 0.5 \\ 0.06 & 0 & -0.02 & 0.3 \\ -0.04 & 0.02 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Example 3.4

Find the effect of a differential rotation of 0.1 rad about the y -axis followed by a differential translation of [0.1, 0, 0.2] on the given frame B :

$$B = \begin{bmatrix} 0 & 0 & 1 & 10 \\ 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Solution As we saw before, the change in the frame can be found by premultiplying the frame with the differential operator. Substituting the given information and multiplying the matrices, we get

For $dx = 0.1, dy = 0, dz = 0.2, \delta x = 0, \delta y = 0.1, \delta z = 0$,

$$\begin{aligned} [dB] &= [\Delta][B] = \begin{bmatrix} 0 & 0 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 \\ -0.1 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 10 \\ 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0.1 & 0 & 0.4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -0.1 & -0.8 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{3.17}$$

3.5 INTERPRETATION OF THE DIFFERENTIAL CHANGE

The matrix dT in Equations (3.14) and (3.15) represents the changes in a frame as a result of differential motions. The elements of this matrix are

$$dT = \begin{bmatrix} dn_x & do_x & da_x & dp_x \\ dn_y & do_y & da_y & dp_y \\ dn_z & do_z & da_z & dp_z \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (3.18)$$

The dB matrix in Equation (3.17) represents the change in the frame B , as shown in Equation (3.18). Thus, each element of the matrix represents the change in the corresponding element of the frame. For example, this means that the frame is moved a differential amount of 0.4 units along the x -axis, no movement along the y -axis, and a differential amount of -0.8 along the z -axis. It has also rotated such that there is no change in its \bar{n} vector, there is a change of 0.1 in the o_x component of \bar{o} vector, and a change of -0.1 in the a_z component of the \bar{a} vector.

Example 3.5

Find the location and the orientation of the frame B of Example 3.4 after the move.

Solution The new location and orientation of the frame can be found by adding the changes to the original values:

$$B_{\text{new}} = B_{\text{original}} + dB = \begin{bmatrix} 0 & 0 & 1 & 10 \\ 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0.1 & 0 & 0.4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -0.1 & -0.8 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.1 & 1 & 10.4 \\ 1 & 0 & 0 & 5 \\ 0 & 1 & -0.1 & 2.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Can you explain how the final result can still be a valid frame?

3.6 DIFFERENTIAL CHANGES BETWEEN FRAMES

The differential operator Δ in Equation (3.15) represents a differential operator relative to the fixed reference frame and is technically ${}^v\Delta$. However, it is possible to define another differential operator, this time relative to the current frame itself, that will enable us to calculate the same changes in the frame. Since the differential operator relative to the frame $(^T\Delta)$ is relative to a current frame, to find the changes in the frame, we must postmultiply the frame by ${}^T\Delta$ (as we did in Chapter 2). The result will be the same, since they both represent the same changes in the frame. Then

$$\begin{aligned}[dT] &= [\Delta][T] = [T][^T\Delta] \\ [T]^{-1}[\Delta][T] &= [T]^{-1}[T][^T\Delta] \\ [^T\Delta] &= [T]^{-1}[\Delta][T].\end{aligned}\tag{3.19}$$

Thus, Equation (3.19) can be used to calculate the differential operator relative to the frame, ${}^T\Delta$. We can multiply the matrices in Equation (3.19) and simplify the result as follows (assuming that the frame T is represented by an $\bar{n}, \bar{o}, \bar{a}, \bar{p}$ matrix):

$$\begin{aligned}T^{-1} &= \begin{bmatrix} n_x & n_y & n_z & -\bar{p} \cdot \bar{n} \\ o_x & o_y & o_z & -\bar{p} \cdot \bar{o} \\ a_x & a_y & a_z & -\bar{p} \cdot \bar{a} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \Delta = \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\ [T^{-1}][\Delta][T] &= {}^T\Delta = \begin{bmatrix} 0 & -{}^T\delta z & {}^T\delta y & {}^Tdx \\ {}^T\delta z & 0 & -{}^T\delta x & {}^Tdy \\ -{}^T\delta y & {}^T\delta x & 0 & {}^Tdz \\ 0 & 0 & 0 & 0 \end{bmatrix}.\end{aligned}\tag{3.20}$$

As you may notice, the ${}^T\Delta$ is *made* to look exactly like the Δ matrix, but all elements are relative to the current frame, where these element are found from the above multiplication of matrices, and are summarized as follows:

$$\begin{aligned}{}^T\delta_x &= \bar{\delta} \cdot \bar{n}, \\ {}^T\delta_y &= \bar{\delta} \cdot \bar{o}, \\ {}^T\delta_z &= \bar{\delta} \cdot \bar{a}, \\ {}^Td_x &= \bar{n} \cdot [(\bar{\delta} \times \bar{p}) + \bar{d}], \\ {}^Td_y &= \bar{o} \cdot [(\bar{\delta} \times \bar{p}) + \bar{d}], \\ {}^Td_z &= \bar{a} \cdot [(\bar{\delta} \times \bar{p}) + \bar{d}].\end{aligned}\tag{3.21}$$

Please see Paul [1] for the derivation of the preceding equations.

Example 3.6

Find ${}^B\Delta$ the for Example 3.4.

Solution We have the following vectors from the given information and we will substitute these values into Equation (3.21) to calculate vectors Bd and ${}^B\delta$:

$$\bar{n} = [0, 1, 0], \quad \bar{o} = [0, 0, 1], \quad \bar{a} = [1, 0, 0], \quad \bar{p} = [10, 5, 3],$$

$$\bar{\delta} = [0, 0, 1, 0], \quad \bar{d} = [0, 1, 0, 0, 2].$$

$$\bar{\delta} \times \bar{p} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ 0 & 0.1 & 0 \\ 10 & 5 & 3 \end{vmatrix} = [0, 3, 0, -1],$$

$$\bar{\delta} \times \bar{p} + \bar{d} = [0, 3, 0, -1] + [0, 1, 0, 0, 2] = [0, 4, 0, -0.8],$$

$$\begin{aligned}\rightarrow^B dx &= \bar{n} \cdot [\bar{\delta} \times \bar{p} + \bar{d}] = 0(0.4) + 1(0) + 0(-0.8) = 0, \\ \rightarrow^B dy &= \bar{o} \cdot [\bar{\delta} \times \bar{p} + \bar{d}] = 0(0.4) + 0(0) + 1(-0.8) = -0.8, \\ \rightarrow^B dz &= \bar{a} \cdot [\bar{\delta} \times \bar{p} + \bar{d}] = 1(0.4) + 0(0) + 0(-0.8) = 0.4, \\ {}^B\delta x &= \bar{\delta} \cdot \bar{n} = 0(0) + 0.1(1) + 0(0) = 0.1, \\ {}^B\delta y &= \bar{\delta} \cdot \bar{o} = 0(0) + 0.1(1) + 0(0) = 0, \\ {}^B\delta z &= \bar{\delta} \cdot \bar{a} = 0(1) + 0.1(0) + 0(0) = 0.\end{aligned}$$

Substituting into Equation (3.20) yields

$$\begin{aligned}{}^B d &= [0, -0.8, 0.4], \quad {}^B \delta = [0.1, 0, 0], \\ {}^B \Delta &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -0.1 & -0.8 \\ 0 & 0.1 & 0 & 0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix}.\end{aligned}$$

As you can see, these values for ${}^B \Delta$ are not the same as Δ . However, postmultiplying the B matrix by ${}^B \Delta$ yields the same result dB as before.

Example 3.7

Calculate ${}^B \Delta$ of Example 3.6 directly from the differential operator.

Solution Using Equation (3.19), we can calculate the ${}^B \Delta$ directly as

$$[{}^B \Delta] = [B^{-1}][\Delta][B] = \begin{bmatrix} 0 & 1 & 0 & -5 \\ 0 & 0 & 1 & -3 \\ 1 & 0 & 0 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 \\ -0.1 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 10 \\ 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -0.1 & -0.8 \\ 0 & 0.1 & 0 & 0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

which, of course, is the same result as in Example 3.6.

3.7 DIFFERENTIAL MOTIONS OF A ROBOT AND ITS HAND FRAME

What we saw in the previous section was the changes made to a frame as a result of differential motions. This only relates to the frame changes, but not how they were accomplished. In this section, we will relate the changes to the mechanism — in this case, the robot that accomplishes the differential motions. We will learn how the robot's movements are translated into the frame changes at the hand.

The frame we discussed previously may be any frame, including the hand frame of a robot. dT describes the changes in the components of the $\bar{n}, \bar{o}, \bar{a}, \bar{p}$ vectors. If the frame were the hand frame of the robot, then we would need to find out how the differential motions of the joints of the robot would relate to differential motions of the hand frame and, specifically, to dT . Of course, this relationship is a function of the robot's configuration and design, but also a function of its instantaneous location and orientation. For example, the simple 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 or not the arm is completely extended, and whether it is pointed in any direction, it would also translate into very different joint velocities to yield the same hand velocity. Of course, as we discussed before, the Jacobian of the robot will create this link between the joint movements and the hand movement as follows:

$$\begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \\ \delta z \end{bmatrix} = \begin{bmatrix} & & & d\theta_1 \\ & & & d\theta_2 \\ Robot & Jacobian & d\theta_3 \\ & & d\theta_4 \\ & & d\theta_5 \\ & & d\theta_6 \end{bmatrix}, \text{ or } [D] = [J][D_\theta]. \quad (3.10)$$

3.8 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 Equation (3.10), we can see that the first element in $[D]$ is dx . This means that 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, and, thus, its derivative will be dx . The same will be true for dy and dz . Considering the $\bar{n}, \bar{o}, \bar{a}, \bar{p}$ matrix, we may pick the corresponding elements of p_x , p_y , and p_z and differentiate them to get dx , dy , and dz .

For example, consider the simple revolute arm of Example 2.19. The last column of the forward kinematic equation of the robot is

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ S_{234}a_4 + S_{23}a_3 + S_2a_2 \\ 1 \end{bmatrix}. \quad (3.22)$$

Taking the derivative of p_x yields

$$p_x = C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2),$$

$$dp_x = \frac{\partial p_x}{\partial \theta_1} d\theta_1 + \frac{\partial p_x}{\partial \theta_2} d\theta_2 + \dots + \frac{\partial p_x}{\partial \theta_6} d\theta_6,$$

$$\begin{aligned} dp_x = & -S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2)d\theta_1 + C_1[-S_{234}a_4 - S_{23}a_3 - S_2a_2]d\theta_2 \\ & + C_1[-S_{234}a_4 - S_{23}a_3]d\theta_3 + C_1[-S_{234}a_4]d\theta_4. \end{aligned}$$

From this, we can write the first row of the Jacobian as

$$\frac{\partial p_x}{\partial \theta_1} = J_{11} = -S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2),$$

$$\begin{aligned}
 \frac{\partial p_x}{\partial \theta_2} &= J_{12} = C_1[-S_{234}a_4 - S_{23}a_3 - S_2a_2], \\
 \frac{\partial p_x}{\partial \theta_3} &= J_{13} = C_1[-S_{234}a_4 - S_{23}a_3], \\
 \frac{\partial p_x}{\partial \theta_4} &= J_{14} = C_1[-S_{234}a_4], \\
 \frac{\partial p_x}{\partial \theta_5} &= J_{15} = 0, \\
 \frac{\partial p_x}{\partial \theta_6} &= J_{16} = 0.
 \end{aligned} \tag{3.23}$$

The same can be done for the next two columns. 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 . As a result, we will 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 will instead use the approach presented next.

Paul [1] has shown that we can write the velocity equation relative to the last frame as

$$[{}^T_6 D] = [{}^T_6 J][D_\theta]. \tag{3.24}$$

This means that for the same joint differential motions, premultiplied with the Jacobian relative to the last frame, you will get the hand differential motions relative to the last frame. Paul [1] has also shown that one can calculate the Jacobian with respect to the last frame using the following simple formulas:

- The differential motion relationship of Equation can be written as

$$\begin{bmatrix} {}^T_6 d_x \\ {}^T_6 d_y \\ {}^T_6 d_z \\ {}^T_6 \delta_x \\ {}^T_6 \delta_y \\ {}^T_6 \delta_z \end{bmatrix} = \begin{bmatrix} {}^T_6 J_{11} & {}^T_6 J_{12} & \dots & {}^T_6 J_{16} \\ {}^T_6 J_{21} & {}^T_6 J_{22} & \dots & {}^T_6 J_{26} \\ {}^T_6 J_{31} & \ddots & \dots & {}^T_6 J_{36} \\ {}^T_6 J_{41} & \ddots & \dots & {}^T_6 J_{46} \\ {}^T_6 J_{51} & \ddots & \dots & {}^T_6 J_{56} \\ {}^T_6 J_{61} & \ddots & \dots & {}^T_6 J_{66} \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \\ \vdots \\ d\theta_5 \\ d\theta_6 \end{bmatrix}.$$

- Assuming that any combination of $A_1, A_2 \dots A_n$ can be expressed with a corresponding $\bar{n}, \bar{o}, \bar{a}, \bar{p}$ matrix, the corresponding elements of the matrix will be used to calculate the Jacobian.
- If joint i under consideration is a revolute joint, then:

$$\begin{aligned}
 {}^T_6 J_{1i} &= (-n_x p_y + n_y p_x), & {}^T_6 J_{2i} &= (-o_x p_y + o_y p_x), & {}^T_6 J_{3i} &= (-a_x p_y + a_y p_x), \\
 {}^T_6 J_{4i} &= n_z, & {}^T_6 J_{5i} &= o_z, & {}^T_6 J_{6i} &= a_z.
 \end{aligned} \tag{3.25}$$

- If joint i under consideration is a prismatic joint, then

$$\begin{aligned} {}^T_6 J_{1i} &= n_z, & {}^T_6 J_{2i} &= o_z, & {}^T_6 J_{3i} &= a_z, \\ {}^T_6 J_{4i} &= 0, & {}^T_6 J_{5i} &= 0, & {}^T_6 J_{6i} &= 0. \end{aligned} \quad (3.26)$$

- For Equations (3.25) and (3.26), for column i use ${}^{i-1}T_6$:

For column 1, use ${}^0T_6 = A_1A_2A_3A_4A_5A_6$.

For column 2, use ${}^1T_6 = A_2A_3A_4A_5A_6$.

For column 3, use ${}^2T_6 = A_3A_4A_5A_6$.

For column 4, use ${}^3T_6 = A_4A_5A_6$.

For column 5, use ${}^4T_6 = A_5A_6$.

For column 6, use ${}^5T_6 = A_6$.

Example 3.8

Using Equation (3.22), find the elements of the second row of the Jacobian for the simple revolute robot.

Solution For the second row of the Jacobian, we will have to differentiate the p_y expression of Equation (3.22) as follows:

$$p_y = S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2),$$

$$dp_y = \frac{\partial p_y}{\partial \theta_1} d\theta_1 + \frac{\partial p_y}{\partial \theta_2} d\theta_2 + \dots + \frac{\partial p_y}{\partial \theta_6} d\theta_6,$$

$$dp_y = C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2)d\theta_1 + S_1[-S_{234}a_4(d\theta_2 + d\theta_3 + d\theta_4)$$

$$-S_{23}a_3(d\theta_2 + d\theta_3) - S_2a_2(d\theta_2)].$$

Rearranging terms yields

$$\frac{\partial p_y}{\partial \theta_1} d\theta_1 = J_{21}d\theta_1 = C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2)d\theta_1,$$

$$\frac{\partial p_y}{\partial \theta_2} d\theta_2 = J_{22}d\theta_2 = S_1(-S_{234}a_4 - S_{23}a_3 - S_2a_2)d\theta_2,$$

$$\frac{\partial p_y}{\partial \theta_3} d\theta_3 = J_{23}d\theta_3 = S_1(-S_{234}a_4 - S_{23}a_3)d\theta_3,$$

$$\frac{\partial p_y}{\partial \theta_4} d\theta_4 = J_{24}d\theta_4 = S_1(-S_{234}a_4)d\theta_4,$$

$$\frac{\partial p_y}{\partial \theta_5} d\theta_5 = J_{25}d\theta_5 = 0, \quad \frac{\partial p_y}{\partial \theta_6} d\theta_6 = J_{26}d\theta_6 = 0.$$

Example 3.9

Find the ${}^T_6 J_{11}$ and ${}^T_6 J_{41}$ elements of the Jacobian for the simple revolute robot.

Solution Since we want to calculate two elements of the first column of the Jacobian, we need to use $A_1 A_2 \dots A_6$ matrix. From Example 2.19, we get

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

Using the corresponding values of $\bar{n}, \bar{o}, \bar{a}, \bar{p}$ and Equation (3.25) for revolute joints, we find that

$$\begin{aligned} {}^T_6 J_{11} &= (-n_x p_y + n_y p_x) \\ &= -[C_1(C_{234}C_5C_6 - S_{234}S_6) - S_1S_5C_6] \times [S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2)] \\ &\quad + [S_1(C_{234}C_5C_6 - S_{234}S_6) + C_1S_5C_6] \times [C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2)] \\ &= S_5C_6(C_{234}a_4 + C_{23}a_3 + C_2a_2), \\ {}^T_6 J_{41} &= n_z = S_{234}C_5C_6 + C_{234}S_6. \end{aligned} \quad (3.28)$$

As you probably notice, the results in Equations (3.23) and (3.28) 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.

3.9 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 that a robot's joints are moved a differential amount. Using Equation (3.10) and knowing the Jacobian we can calculate the $[D]$ matrix, which contains values of $dx, dy, dz, \delta x, \delta y, \delta z$ (differential motions of the hand). These can be substituted in Equation (3.16) to form the differential operator. Next, Equation (3.15) can be used to calculate dT . This can be used to locate the new position and orientation of the robot's hand. Thus, the differential motions of the robot's joints are ultimately related to the hand frame of the robot.

Alternatively, Equation (3.24), and the Jacobian can be used to calculate the ${}^T_6 D$ matrix, which contains values of ${}^T_6 dx, {}^T_6 dy, {}^T_6 dz, {}^T_6 \delta x, {}^T_6 \delta y, {}^T_6 \delta z$, (differential motions of the hand relative to the current frame). These can be substituted in Equation (3.20) to form the differential operator. Next, Equation (3.19) can be used to calculate dT as before.

Example 3.10

The hand frame of a robot with five degrees of freedom, its numerical Jacobian for this instant, 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:

$$T_6 = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad J = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ -2 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} d\theta_1 \\ d\theta_2 \\ ds_1 \\ d\theta_4 \\ d\theta_5 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.1 \\ 0.05 \\ 0.1 \\ 0 \end{bmatrix}.$$

Solution It is assumed that the robot can only rotate about the x - and y -axes, since it has only five degrees of freedom. Using Equation (3.10), we can calculate the $[D]$ matrix, which is then substituted into Equation (3.16) as follows:

$$D = \begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \end{bmatrix} = JD_\theta = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ -2 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.1 \\ 0.05 \\ 0.1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.3 \\ -0.15 \\ -0.4 \\ 0 \\ -0.1 \end{bmatrix} \rightarrow \Delta = \begin{bmatrix} 0 & 0 & -0.1 & 0.3 \\ 0 & 0 & 0 & -0.15 \\ 0.1 & 0 & 0 & -0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

From Equation (3.15), we get

$$[dT_6] = [\Delta][T_6] = \begin{bmatrix} 0 & 0 & -0.1 & 0.3 \\ 0 & 0 & 0 & -0.15 \\ 0.1 & 0 & 0 & -0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -0.1 & 0 & 0.1 \\ 0 & 0 & 0 & -0.15 \\ 0.1 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The new location of the frame after the differential motion is

$$T_6 = dT_6 + T_{6\text{ORIGINAL}} = \begin{bmatrix} 0 & -0.1 & 0 & 0.1 \\ 0 & 0 & 0 & -0.15 \\ 0.1 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -0.1 & 0 & 5.1 \\ 0 & 0 & -1 & 2.85 \\ 0.1 & 1 & 0 & 2.1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3.10 INVERSE JACOBIAN

To calculate the differential motions (or velocities) needed at the joints of the robot for a desired hand differential motion (or velocity), we need to calculate the inverse of the Jacobian and use it in the following equation:

$$\begin{aligned} [D] &= [J][D_\theta] \\ [J^{-1}][D] &= [J^{-1}][J][D_\theta] \quad \rightarrow \quad [D_\theta] = [J^{-1}][D] \end{aligned} \quad (3.29)$$

and, similarly,

$$[{}^T J^{-1}] [{}^T D] = [{}^T J^{-1}] [{}^T J] [D_\theta] \rightarrow D_\theta [{}^T J^{-1}] [{}^T D]. \quad (3.30)$$

This means that knowing the inverse of the Jacobian, one can calculate how fast each joint must move so that the robot's hand will yield a desired differential motion or velocity. In reality, this analysis, and not the forward differential motion calculations, is the main purpose of the differential motion analysis. Imagine that there is a robot that is laying glue on a plate. It is necessary that not only must the robot follow a particular path on a flat plane, but it must also 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.

As was mentioned 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. Thus, it is necessary to calculate the Jacobian's numerical values continuously. This means that in order to be able to calculate enough number of joint velocities per second to have accurate velocities, the process must be very efficient and quick; otherwise, the results will be inaccurate and useless.

Inverting the Jacobian may be done in two ways, both of which are very difficult, computationally intensive, and time consuming. One technique is to find the symbolic inverse of the Jacobian and then substitute the values into it to compute the velocities. The other technique is to substitute the numbers in the Jacobian and then invert the numerical matrix through techniques such as Gaussian elimination or other similar approaches. Although these approaches are both possible, they are not usually done.

Instead, one may use the inverse kinematic equations to calculate the joint velocities. Consider Equation (2.62) (repeated here), which yields the value of θ_1 for the simple revolute robot:

$$p_x S_1 - p_y C_1 = 0 \rightarrow \theta_1 = \tan^{-1} \left(\frac{p_y}{p_x} \right) \text{ and } \theta_1 = \theta_1 + 180^\circ. \quad (2.62)$$

We can differentiate the relationship to find $d\theta_1$, which is the differential value of θ_1 :

$$\begin{aligned} p_y S_1 &= p_y C_1, \\ dp_y S_1 + p_y C_1 d\theta_1 &= dp_y C_1 - dp_y S_1 d\theta_1, \\ d\theta_1 (p_x C_1 + p_y S_1) &= -dp_x S_1 + dp_y C_1, \\ d\theta_1 &= \frac{-dp_x S_1 + dp_y C_1}{(p_x C_1 + p_y S_1)}. \end{aligned} \quad (3.31)$$

Similarly, from Equation (2.68) (repeated here), we get

$$\begin{aligned}
 S_{234}(C_1 a_x + S_1 a_y) &= C_{234} a_z, \\
 C_{234}(d\theta_2 + d\theta_3 + d\theta_4)(C_1 a_x + S_1 a_y) + S_{234}[-a_x S_1 d\theta_1 + C_1 da_x + a_y C_1 d\theta_1 + S_1 da_y] \\
 &= -S_{234}(d\theta_2 + d\theta_3 + d\theta_4)a_z + C_{234}da_z, \\
 (d\theta_2 + d\theta_3 + d\theta_4) &= \frac{S_{234}[a_x S_1 d\theta_1 - C_1 da_x - a_y C_1 d\theta_1 - S_1 da_y] + C_{234}da_z}{C_{234}(C_1 a_x + S_1 a_y) + S_{234}a_z}.
 \end{aligned} \tag{3.32}$$

Equation (3.32) gives the combination of three differential motions in terms of known values. Remember that da_x , da_y , etc., are all known from the dT matrix, since dT is the differential change of the $\bar{n}, \bar{o}, \bar{a}, \bar{p}$ matrix:

$$dT = \begin{bmatrix} dn_x & do_x & da_x & dp_x \\ dn_y & do_y & da_y & dp_y \\ dn_z & do_z & da_z & dp_z \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.33}$$

Next we can differentiate Equation (2.64) to find a relationship for $d\theta_3$ as follows:

$$\begin{aligned}
 2a_2 a_3 C_3 &= (p_x C_1 + p_y S_1 - C_{234} a_4)^2 + (p_z - S_{234} a_4)^2 - a_2^2 - a_3^2, \\
 -2a_2 a_3 S_3 d\theta_3 &= 2(p_x C_1 + p_y S_1 - C_{234} a_4) \\
 &\quad \times [C_1 dp_x - p_x S_1 d\theta_1 + S_1 dp_y + p_y C_1 d\theta_1 \\
 &\quad + a_4 S_{234}(d\theta_2 + d\theta_3 + d\theta_4)] \\
 &\quad + 2(p_z - S_{234} a_4)[dp_z - a_4 C_{234}(d\theta_2 + d\theta_3 + d\theta_4)].
 \end{aligned} \tag{3.34}$$

Although Equation (3.34) is long, all elements in it are already known, and $d\theta_3$ can be calculated.

Next, differentiating Equation (2.70), we get

$$\begin{aligned}
 S_2[(C_3 a_3 + a_2)^2 + S_3^2 a_3^2] &= (C_3 a_3 + a_2)(p_z - S_{234} a_4) - S_3 a_3(p_x C_1 + p_y S_1 - C_{234} a_4), \\
 C_2 d\theta_2 [(C_3 a_3 + a_2)^2 + S_3^2 a_3^2] + S_2[2(C_3 a_3 + a_2)(-a_3 S_3 d\theta_3) + 2a_3^2 S_3 C_3 d\theta_3] \\
 &= -a_3 S_3 d\theta_3(p_z - S_{234} a_4) + (C_3 a_3 + a_2)[dp_z - a_4 C_{234}(d\theta_2 + d\theta_3 + d\theta_4)] \tag{3.35} \\
 &\quad -a_3 C_3 d\theta_3(p_x C_1 + p_y S_1 - C_{234} a_4) \\
 &\quad -S_3 a_3[dp_x C_1 - p_x S_1 d\theta_1 + dp_y S_1 + p_y C_1 d\theta_1 + S_{234} a_4(d\theta_2 + d\theta_3 + d\theta_4)],
 \end{aligned}$$

which will yield $d\theta_2$, since all other elements are known. This will also enable us to calculate $d\theta_4$ from Equation (3.32). Next we will differentiate C_5 from Equation (2.73) to get

$$\begin{aligned}
 C_5 &= -C_1 a_y + S_1 a_x, \\
 -S_5 d\theta_5 &= S_1 a_y d\theta_1 - C_1 da_y + C_1 a_x d\theta_1 + S_1 da_x,
 \end{aligned} \tag{3.36}$$

which results in $d\theta_5$. Lastly, we will differentiate the 2,1 elements of Equation (2.75) to calculate $d\theta_6$:

$$\begin{aligned} S_6 &= -S_{234}(C_1 n_x + S_1 n_y) + C_{234} n_z, \\ C_6 d\theta_6 &= -C_{234}(C_1 n_x + S_1 n_y)(d\theta_2 + d\theta_3 + d\theta_4) \\ &\quad - S_{234}(-S_1 n_x d\theta_1 + C_1 d n_x + C_1 n_y d\theta_1 + S_1 d n_y) \\ &\quad - S_{234} n_z(d\theta_2 + d\theta_3 + d\theta_4) + C_{234} d n_z. \end{aligned} \quad (3.37)$$

As you can see, there are six differential equations that result in six differential joint values, from which velocities can be calculated. The robot controller may be programmed by these six equations, which enable the controller to quickly calculate velocities and run the robot joints accordingly.

Example 3.11

The revolute robot of Example 2.19 is in the configuration shown in Figure 3.3. 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:

$$dx/dt = 1 \text{ in/sec}, \quad dy/dt = -2 \text{ in/sec}, \quad \delta x/dt = 0.1 \text{ rad/sec},$$

$$\theta_1 = 0^\circ, \theta_2 = 90^\circ, \theta_3 = 0^\circ, \theta_4 = 90^\circ, \theta_5 = 0^\circ, \theta_6 = 45^\circ$$

$$a_2 = 15'', a_3 = 15'', a_4 = 15''.$$

TABLE 3.1 PARAMETERS FOR THE ROBOT
OF EXAMPLE 2.19

#		<i>d</i>	<i>a</i>	
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

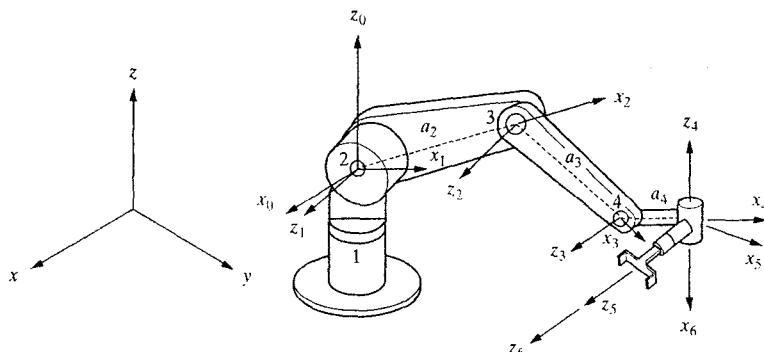


Figure 3.3 Reference frames for the simple six-degree-of-freedom articulate robot.

Solution First, we will substitute these values into Equation (2.57) (repeated here) to obtain the final position and orientation of the robot. Please notice that the actual position and orientation of the robot does depend on what is 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, we find that

$${}^R T_H = A_1 A_2 A_3 A_4 A_5 A_6 =$$

$$\begin{bmatrix} C_1(C_{234}C_5C_6 - S_{234}S_6) & C_1(-C_{234}C_5C_6 - S_{234}C_6) & C_1(C_{234}S_5) & C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ -S_1S_5C_6 & +S_1S_5S_6 & +S_1C_5 & \\ S_1(C_{234}C_5C_6 - S_{234}S_6) & S_1(-C_{234}C_5C_6 - S_{234}C_6) & S_1(C_{234}S_5) & S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ +C_1S_5C_6 & -C_1S_5S_6 & -C_1C_5 & \\ S_{234}C_5C_6 + C_{234}S_6 & -S_{234}C_5C_6 + C_{234}C_6 & S_{234}S_5 & S_{234}a_4 + S_{23}a_3 + S_2a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$${}^R T_H = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.707 & 0.707 & 0 & -5 \\ 0 & 0 & -1 & 0 \\ -0.707 & -0.707 & 0 & 30 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Substituting the desired differential motion values into Equations (3.15) and (3.16), we get

$$\Delta = \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -0.1 & -2 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$[dT] = [\Delta][T] = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.0707 & 0.0707 & 0 & -5 \\ 0 & 0 & -0.1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Substituting the values from the dT and T matrices into Equation (3.31), 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 foregoing value for θ_1 causes a degenerate condition for the robot, and, as a result, other angular velocities cannot be calculated for this configuration.

3.11 DESIGN PROJECT

This is a continuation of the design project which we started in Chapter 2. If you did design a three-degree-of-freedom robot and 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 three-

degree-of-freedom robot, the calculations are relatively easy. Remember that with three degrees of freedom, 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 dP_x , dP_y , and dP_z . 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), then 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 subsequent chapters.

3.12 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 topics together. Through this, we found how to 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 also 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, one can control both the motions and the velocity of a multiple-degree-of-freedom robot in space. We can also follow the location of the hand frame as it moves in space.

In the next chapter, we will continue with the derivation of dynamic equations of motion, and we will be able to 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. Craig, John J., *Introduction to Robotics: Mechanics and Control*, 2d Edition, Addison Wesley, 1989.
3. Shahinpoor, Mohsen, *A Robot Engineering Textbook*, Harper & Row, 1987.
4. Koren, Yoram, *Robotics for Engineers*, McGraw-Hill, 1985.
5. Fu, K. S., R.C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill, New York, 1987.
6. Asada, Haruhiko, J.J. E. Slotine, *Robot Analysis and Control*, John Wiley and Sons, New York, 1986.
7. Sciavicco, Lorenzo, B. Siciliano, *Modeling and Control of Robot Manipulators*, McGraw-Hill, New York, 1996.

PROBLEMS

1. Suppose that the location and orientation of a hand frame is expressed by the accompanying 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:

$${}^R T_H = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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 .

$$T = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 0 & 1 & 3 \\ 0 & -1 & 0 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad dT = \begin{bmatrix} 0 & -0.1 & -0.1 & 0.6 \\ 0.1 & 0 & 0 & 0.5 \\ -0.1 & 0 & 0 & -0.5 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

3. Suppose that 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 ?

$$A = \begin{bmatrix} 0 & 0 & 1 & 10 \\ 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

4. The initial location and orientation of a robot's hand is given by T_1 , and its new location and orientation after a change is given by T_2 .

- Find a transformation matrix Q that will accomplish this transform (in Universe frame).
- Assuming that 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. 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 corresponding Δ :

$$T_6 = \begin{bmatrix} 0 & 1 & 0 & 10 \\ 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^T_6 J = \begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad D_\theta = \begin{bmatrix} 0 \\ 0.1 \\ -0.1 \\ 0.2 \\ 0.2 \\ 0 \end{bmatrix}.$$

6. Calculate the ${}^T_6 J_{21}$ element of the Jacobian for the revolute robot of Example 2.19.
7. Calculate the ${}^T_6 J_{16}$ element of the Jacobian for the revolute robot of Example 2.19.
8. Using Equation (2.33), differentiate proper elements of the matrix to develop a set of symbolic equations for joint differential motions of a cylindrical robot, and write the corresponding Jacobian.
9. Using Equation (2.35), differentiate proper elements of the matrix to develop a set of symbolic equations for joint differential motions of a spherical robot and write the corresponding Jacobian.
10. 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 given the following:
 $\dot{r} = 0.1 \text{ in/sec}, \dot{\alpha} = 0.05 \text{ rad/sec}, \dot{l} = 0.2 \text{ in/sec}, r = 15 \text{ in}, \alpha = 30^\circ, l = 10 \text{ in.}$
11. 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 given the following:
 $\dot{r} = 2 \text{ in/sec}, \dot{\beta} = 0.05 \text{ rad/sec}, \dot{\gamma} = 0.1 \text{ rad/sec}, r = 20 \text{ in}, \beta = 60^\circ, \gamma = 30^\circ.$
12. 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:
 $\dot{x} = 1 \text{ in/sec}, \dot{y} = 3 \text{ in/sec}, \dot{z} = 5 \text{ in/sec}, \alpha = 45^\circ, r = 20 \text{ in}, l = 25 \text{ in.}$
13. 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:
 $\dot{x} = 5 \text{ in/sec}, \dot{y} = 9 \text{ in/sec}, \dot{z} = 6 \text{ in/sec}, \beta = 60^\circ, r = 20 \text{ in}, \gamma = 30^\circ.$

4

Dynamic Analysis and Forces

4.1 INTRODUCTION

In previous chapters, we studied the kinematic position and differential motions of robots. In this chapter, we will look at the dynamics of robots as it relates to accelerations, loads, and masses and inertias. We will also study the static force relationships of robots.

As you remember from your dynamics course, 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 4.1), as in

$$\sum \bar{F} = m \cdot \bar{a} \quad \text{and} \quad \sum \bar{T} = I \cdot \bar{\alpha}. \quad (4.1)$$

To be able to accelerate a robot's links, it is necessary to have actuators that are capable of exerting large enough forces and torques on the links and joints to move them at a desired acceleration and velocity. Otherwise, the link may not be moving fast enough, and thus the robot will lose its positional accuracy. To be able to calculate how strong each actuator must be, it is necessary to determine the dynamic relationships that govern the *motions of the robot*. These equations are the force–mass–acceleration and the torque–inertia–angular acceleration relationships. 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 and thereby design the actuators to be able to 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 knowing the forces and torques, one can figure out how a mechanism will move. However, in our case, we have already found the equations of motions; besides, it is practically impossible to solve the dynamic equations

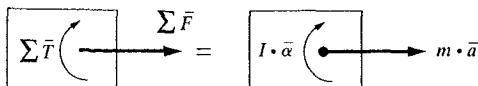


Figure 4.1 Force–mass–acceleration and torque–inertia–angular acceleration relationships for a rigid body.

of robots in all but the simplest cases. Instead, we will 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 see the effects of different inertial loads on the robot and, depending on the desired accelerations, whether certain loads are important. For example, consider a robot in space. Although objects are weightless in space, they do have inertia. As a result, the weight of objects that a robot in space may handle may be trivial, but its inertia is not. So 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 robot used with the Space Shuttle program is very slender, but handles 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 three-dimensional, multiple-degree-of-freedom mechanisms with distributed masses, it is very difficult to use Newtonian mechanics. Instead, one may opt to use other techniques such as Lagrangian mechanics. Lagrangian mechanics is based on energy terms only and thus in many cases is easier to use. Although Newtonian mechanics, as well as other techniques, can be used for this derivation, most references are based on Lagrangian mechanics. In this chapter, we will briefly study Lagrangian mechanics with some examples, and then we will see how it can be used to solve for robot equations. Since this course is primarily intended for undergraduate students, the equations will not be completely derived, but only the results will be demonstrated and discussed. Interested students are encouraged to refer to other references for more detail [1,2,3,4,5,6,7].

4.2 LAGRANGIAN MECHANICS: A SHORT OVERVIEW

Lagrangian mechanics is based on the differentiation of the energy terms with respect to the system's variables and time, as shown next. 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. The Lagrangian mechanics is based on the following two generalized equations, one for linear motions, one for rotational motions. First we will define a Lagrangian as

$$L = K - P, \quad (4.2)$$

where L is the Lagrangian, K is the kinetic energy of the system, and P is the potential energy of the system. Then

$$F_i = \frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{x}_i} \right) - \frac{\partial L}{\partial x_i}, \quad (4.3)$$

$$T_i = \frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{\theta}_i} \right) - \frac{\partial L}{\partial \theta_i}, \quad (4.4)$$

where F is the summation of all external forces for a linear motion, T is the summation of all torques in a rotational motion, and θ and x are system variables. As a result, to get the equations of motion, we need to derive energy equations for the system, and then differentiate the Lagrangian according to Equations (4.3) and (4.4). The following three examples demonstrate the application of Lagrangian mechanics in deriving equations of motion. Notice how the complexity of the terms increases as the number of degrees of freedom (and variables) increases.

Example 4.1

Derive the force-acceleration relationship for the one-degree-of-freedom system shown in Figure 4.2 using both the Lagrangian mechanics, as well as the Newtonian mechanics. Assume that the wheels have negligible inertia.

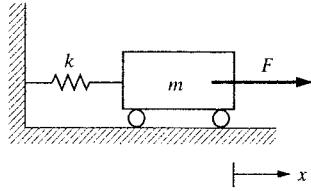


Figure 4.2 Schematic of a simple cart-spring system.

Solution The x -axis denotes the motion of the cart and is used as the variable in this system. Since this is a one-degree-of-freedom system, there will be only one equation describing the motion. Since the motion is linear, we will only use Equation (4.3), as follows:

$$K = \frac{1}{2}mv^2 = \frac{1}{2}m\dot{x}^2 \quad \text{and} \quad P = \frac{1}{2}kx^2 \rightarrow L = K - P = \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2.$$

The derivatives of the Lagrangian are

$$\frac{\partial L}{\partial \dot{x}} = m\ddot{x}, \quad \frac{d}{dt}(m\dot{x}) = m\ddot{x}, \quad \text{and} \quad \frac{\partial L}{\partial x} = -kx.$$

Thus, the equation of motion for the cart is

$$F = m\ddot{x} + kx.$$

To solve the problem with Newtonian mechanics, we will draw the free-body diagram of the cart (Figure 4.3) and will solve for forces as follows:

$$\sum F = ma,$$

$$F - kx = ma \rightarrow F = ma + kx.$$

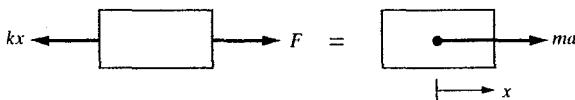


Figure 4.3 Free-body diagram for the spring–cart system.

This is exactly what we expected. For this simple system, it appears that Newtonian mechanics is simpler.

Example 4.2

Derive the equations of motion for the two-degree-of-freedom system shown in Figure 4.4.

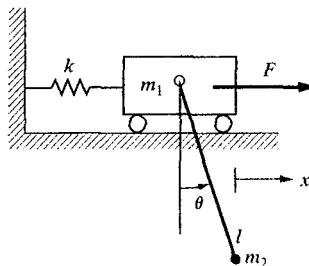


Figure 4.4 Schematic of a cart–pendulum system.

Solution In this problem, there are two degrees of freedom, two coordinates x and θ , and there will be 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 is comprised of the kinetic energy of the cart and of 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

$$V_p^2 = (\dot{x} + l\dot{\theta} \cos \theta)^2 + (l\dot{\theta} \sin \theta)^2.$$

Thus,

$$K = K_{\text{cart}} + K_{\text{pendulum}},$$

$$K_{\text{cart}} = \frac{1}{2}m_1\dot{x}^2$$

$$K_{\text{pendulum}} = \frac{1}{2}m_2(\dot{x} + l\dot{\theta} \cos \theta)^2 + \frac{1}{2}m_2(l^2\dot{\theta}^2 + 2l\dot{\theta}\dot{x} \cos \theta),$$

$$K = \frac{1}{2}(m_1 + m_2)\dot{x}^2 + \frac{1}{2}m_2(l^2\dot{\theta}^2 + 2l\dot{\theta}\dot{x} \cos \theta).$$

Likewise, the potential energy is the summation of the potential energy in the spring and in the pendulum, or

$$P = \frac{1}{2} kx^2 + m_2 gl(1 - \cos \theta).$$

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 equation of motion related to the linear motion are

$$\frac{\partial L}{\partial \dot{x}} = (m_1 + m_2)\ddot{x} + m_2l\dot{\theta}\cos\theta,$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}} \right) = (m_1 + m_2)\ddot{x} + m_2l\ddot{\theta}\cos\theta - m_2l\dot{\theta}^2\sin\theta,$$

$$\frac{\partial L}{\partial x} = -kx,$$

$$F = (m_1 + m_2)\ddot{x} + m_2l\ddot{\theta}\cos\theta - m_2l\dot{\theta}^2\sin\theta + kx.$$

For the rotational motion, it is

$$\frac{\partial L}{\partial \dot{\theta}} = m_2l^2\ddot{\theta} + m_2l\dot{x}\cos\theta,$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) = m_2l^2\ddot{\theta} + m_2l\ddot{x}\cos\theta - m_2l\dot{x}\dot{\theta}\sin\theta,$$

$$\frac{\partial L}{\partial \theta} = -m_2gl\sin\theta - m_2l\dot{\theta}\dot{x}\sin\theta,$$

$$T = m_2l^2\ddot{\theta} + m_2l\ddot{x}\cos\theta + m_2gl\sin\theta.$$

If we write the two equations of motion in a matrix form, we get

$$F = (m_1 + m_2)\ddot{x} + m_2l\ddot{\theta}\cos\theta - m_2l\dot{\theta}^2\sin\theta + kx,$$

$$T = m_2l^2\ddot{\theta} + m_2l\ddot{x}\cos\theta + m_2gl\sin\theta, \quad (4.5)$$

$$\begin{bmatrix} F \\ T \end{bmatrix} = \begin{bmatrix} m_1 + m_2 & m_2l\cos\theta \\ m_2l\cos\theta & m_2l^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & m_2l\sin\theta \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}^2 \\ \dot{\theta}^2 \end{bmatrix} + \begin{bmatrix} kx \\ m_2gl\sin\theta \end{bmatrix}.$$

Example 4.3

Derive the equations of motion for the two-degree-of-freedom system shown in Figure 4.5.

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 that

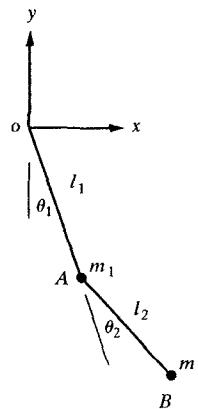


Figure 4.5 A two-link mechanism with concentrated masses.

there are only two degrees of freedom. However, in this example, we will see many more acceleration terms, such as we would expect to see with robots, such as centripetal accelerations and Coriolis accelerations.

We will follow the same format as before. First, we calculate the kinetic and potential energies of the system:

$$K = K_1 + K_2,$$

where

$$K_1 = \frac{1}{2} m_1 l_1^2 \dot{\theta}_1^2.$$

To calculate K_2 , first we will write the position equation for m_2 and then differentiate it for the velocity of m_2 :

$$\begin{cases} x_2 = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) = l_1 S_1 + l_2 S_{12}, \\ y_2 = -l_1 C_1 - l_2 C_{12}, \\ \dot{x}_2 = l_1 C_1 \dot{\theta}_1 + l_2 C_{12}(\dot{\theta}_1 + \dot{\theta}_2), \\ \dot{y}_2 = l_1 S_1 \dot{\theta}_1 + l_2 S_{12}(\dot{\theta}_1 + \dot{\theta}_2). \end{cases}$$

Since $V^2 = \dot{x}^2 + \dot{y}^2$, we get

$$\begin{aligned} V_2^2 &= l_1^2 \dot{\theta}_1^2 + l_2^2 (\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1 \dot{\theta}_2) + 2l_1 l_2 (\dot{\theta}_1^2 + \dot{\theta}_1 \dot{\theta}_2)(C_1 C_{12} + S_1 S_{12}) \\ &= l_1^2 \dot{\theta}_1^2 + l_2^2 (\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1 \dot{\theta}_2) + 2l_1 l_2 C_2 (\dot{\theta}_1^2 + \dot{\theta}_1 \dot{\theta}_2). \end{aligned}$$

Then the kinetic energy for the second mass is

$$K_2 = \frac{1}{2} m_2 l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2 (\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1 \dot{\theta}_2) + m_2 l_1 l_2 C_2 (\dot{\theta}_1^2 + \dot{\theta}_1 \dot{\theta}_2),$$

and the total kinetic energy is

$$K = \frac{1}{2} (m_1 + m_2) l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2 (\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1 \dot{\theta}_2) + m_2 l_1 l_2 C_2 (\dot{\theta}_1^2 + \dot{\theta}_1 \dot{\theta}_2).$$

The potential energy of the system can be written as

$$P_1 = -m_1 g l_1 C_1,$$

$$P_2 = -m_2 g l_1 C_1 - m_2 g l_2 C_{12},$$

$$P = P_1 + P_2 = -(m_1 + m_2) g l_1 C_1 - m_2 g l_2 C_{12}.$$

Notice that in this case, the datum (zero potential energy) is chosen at the axis of rotation “o.”

The Lagrangian for the system is

$$\begin{aligned} L = K - P &= \frac{1}{2} (m_1 + m_2) l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2 (\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1 \dot{\theta}_2) \\ &\quad + m_2 l_1 l_2 C_2 (\dot{\theta}_1^2 + \dot{\theta}_1 \dot{\theta}_2) + (m_1 + m_2) g l_1 C_1 + m_2 g l_2 C_{12}. \end{aligned}$$

The derivatives of the Lagrangian are

$$\frac{\partial L}{\partial \dot{\theta}_1} = (m_1 + m_2) l_1^2 \dot{\theta}_1 + m_2 l_2^2 (\dot{\theta}_1 + \dot{\theta}_2) + 2m_2 l_1 l_2 C_2 \dot{\theta}_1 + m_2 l_1 l_2 C_2 \dot{\theta}_2,$$

$$\begin{aligned} \frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_1} &= [(m_1 + m_2) l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 C_2] \ddot{\theta}_1 + [m_2 l_2^2 + m_2 l_1 l_2 C_2] \ddot{\theta}_2 \\ &\quad - 2m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2 - m_2 l_1 l_2 S_2 \dot{\theta}_2^2, \end{aligned}$$

$$\frac{\partial L}{\partial \theta_1} = -(m_1 + m_2) g l_1 S_1 - m_2 g l_2 S_{12}.$$

From Equation (4.4), the first equation of motion is

$$\begin{aligned} T_1 &= [(m_1 + m_2) l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 C_2] \ddot{\theta}_1 + [m_2 l_2^2 + m_2 l_1 l_2 C_2] \ddot{\theta}_2 \\ &\quad - 2m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2 - m_2 l_1 l_2 S_2 \dot{\theta}_2^2 + (m_1 + m_2) g l_1 S_1 + m_2 g l_2 S_{12}. \end{aligned}$$

Similarly,

$$\frac{\partial L}{\partial \dot{\theta}_2} = m_2 l_2^2 (\dot{\theta}_1 + \dot{\theta}_2) + m_2 l_1 l_2 C_2 \dot{\theta}_1,$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_2} = m_2 l_2^2 (\ddot{\theta}_1 + \ddot{\theta}_2) + m_2 l_1 l_2 C_2 \ddot{\theta}_1 - m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2,$$

$$\frac{\partial L}{\partial \theta_2} = -m_2 l_1 l_2 S_2 (\dot{\theta}_1^2 + \dot{\theta}_1 \dot{\theta}_2) - m_2 g l_2 S_{12},$$

$$T_2 = (m_2 l_2^2 + m_2 l_1 l_2 C_2) \ddot{\theta}_1 + m_2 l_2^2 \ddot{\theta}_2 + m_2 l_1 l_2 S_2 \dot{\theta}_1^2 + m_2 g l_2 S_{12}.$$

Writing these two equations in a matrix form, we get

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} (m_1 + m_2)l_1^2 + m_2l_2^2 + 2m_2l_1l_2C_2 & m_2l_2^2 + m_2l_1l_2C_2 \\ m_2l_2^2 + m_2l_1l_2C_2 & m_2l_2^2 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & -m_2l_1l_2S_2 \\ m_2l_1l_2S_2 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \end{bmatrix} + \begin{bmatrix} -m_2l_1l_2S_2 & -m_2l_1l_2S_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1\dot{\theta}_2 \\ \dot{\theta}_2\dot{\theta}_1 \end{bmatrix} \quad (4.6)$$

$$+ \begin{bmatrix} (m_1 + m_2)gl_1S_1 + m_2gl_2S_{12} \\ m_2gl_2S_{12} \end{bmatrix}.$$

Example 4.4

Using the Lagrangian method, derive the equations of motion for the two-degree-of-freedom robot arm, shown in Figure 4.6. The center of mass for each link is at the center of the link. The moments of inertia are I_1 and I_2 .

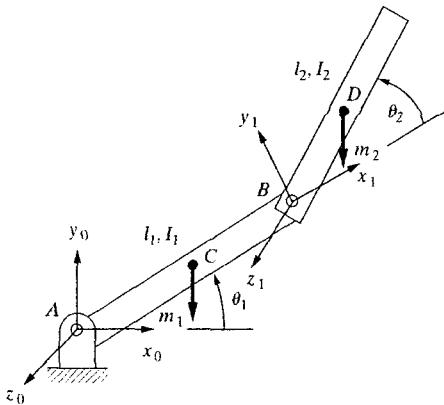


Figure 4.6 A two-degree-of-freedom robot arm.

Solution The solution of this example robot arm is in fact similar to the solution of Example 4.3. 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 will follow the same steps as before. First, we calculate the velocity of the center of mass of link 2 by differentiating its position:

$$x_D = l_1C_1 + 0.5l_2C_{12} \rightarrow \dot{x}_D = -l_1S_1\dot{\theta}_1 - 0.5l_2S_{12}(\dot{\theta}_1 + \dot{\theta}_2),$$

$$y_D = l_1S_1 + 0.5l_2S_{12} \rightarrow \dot{y}_D = l_1C_1\dot{\theta}_1 + 0.5l_2C_{12}(\dot{\theta}_1 + \dot{\theta}_2).$$

Therefore, the total velocity of the center of mass of link 2 is

$$\begin{aligned} V_D^2 &= \dot{x}_D^2 + \dot{y}_D^2 \\ &= \dot{\theta}_1^2(l_1^2 + 0.25l_2^2 + l_1l_2C_2) + \dot{\theta}_2^2(0.25l_2^2) + \dot{\theta}_1\dot{\theta}_2(0.5l_2^2 + l_1l_2C_2). \end{aligned} \quad (4.7)$$

The kinetic energy of the total system is the sum of the kinetic energies of links 1 and 2. Remembering the formula for finding kinetic energy for a link rotating about a fixed axis (for link 1) and about the center of mass (for link 2), we have

$$\begin{aligned}
 K = K_1 + K_2 &= \left[\frac{1}{2} I_A \dot{\theta}_1^2 \right] + \left[\frac{1}{2} I_D (\dot{\theta}_1 + \dot{\theta}_2)^2 + \frac{1}{2} m_2 V_D^2 \right] \\
 &= \left[\frac{1}{2} \left(\frac{1}{3} m_1 l_1^2 \right) \dot{\theta}_1^2 \right] + \left[\frac{1}{2} \left(\frac{1}{12} m_2 l_2^2 \right) (\dot{\theta}_1 + \dot{\theta}_2)^2 + \frac{1}{2} m_2 V_D^2 \right]. \tag{4.8}
 \end{aligned}$$

Substituting Equation (4.7) into Equation (4.8) and regrouping, we get

$$\begin{aligned}
 K &= \dot{\theta}_1^2 \left(\frac{1}{6} m_1 l_1^2 + \frac{1}{6} m_2 l_2^2 + \frac{1}{2} m_2 l_1^2 + \frac{1}{2} m_2 l_1 l_2 C_2 \right) \\
 &\quad + \dot{\theta}_2^2 \left(\frac{1}{6} m_2 l_2^2 \right) + \dot{\theta}_1 \dot{\theta}_2 \left(\frac{1}{3} m_2 l_2^2 + \frac{1}{2} m_2 l_1 l_2 C_2 \right). \tag{4.9}
 \end{aligned}$$

The potential energy of the system is the sum of the potential energies of the two links:

$$P = m_1 g \frac{l_1}{2} S_1 + m_2 g \left(l_1 S_1 + \frac{l_2}{2} S_{12} \right). \tag{4.10}$$

The Lagrangian for the two-link robot arm is

$$\begin{aligned}
 L = K - P &= \dot{\theta}_1^2 \left(\frac{1}{6} m_1 l_1^2 + \frac{1}{6} m_2 l_2^2 + \frac{1}{2} m_2 l_1^2 + \frac{1}{2} m_2 l_1 l_2 C_2 \right) + \dot{\theta}_2^2 \left(\frac{1}{6} m_2 l_2^2 \right) \\
 &\quad + \dot{\theta}_1 \dot{\theta}_2 \left(\frac{1}{3} m_2 l_2^2 + \frac{1}{2} m_2 l_1 l_2 C_2 \right) - m_1 g \frac{l_1}{2} S_1 - m_2 g \left(l_1 S_1 + \frac{l_2}{2} S_{12} \right).
 \end{aligned}$$

Taking the derivatives of the Lagrangian and substituting the terms into Equation (4.4) yields the following two equations of motion:

$$\begin{aligned}
 T_1 &= \left(\frac{1}{3} m_1 l_1^2 + m_2 l_1^2 + \frac{1}{3} m_2 l_2^2 + m_2 l_1 l_2 C_2 \right) \ddot{\theta}_1 + \left(\frac{1}{3} m_2 l_2^2 + \frac{1}{2} m_2 l_1 l_2 C_2 \right) \ddot{\theta}_2 \\
 &\quad - (m_2 l_1 l_2 S_2) \dot{\theta}_1 \dot{\theta}_2 - \left(\frac{1}{2} m_2 l_1 l_2 S_2 \right) \dot{\theta}_2^2 \\
 &\quad + \left(\frac{1}{2} m_1 + m_2 \right) g l_1 C_1 + \frac{1}{2} m_2 g l_2 C_{12}, \tag{4.11}
 \end{aligned}$$

$$\begin{aligned}
 T_2 &= \left(\frac{1}{3} m_2 l_2^2 + \frac{1}{2} m_2 l_1 l_2 C_2 \right) \ddot{\theta}_1 + \left(\frac{1}{3} m_2 l_2^2 \right) \ddot{\theta}_2 + \left(\frac{1}{2} m_2 l_1 l_2 S_2 \right) \dot{\theta}_1^2 \\
 &\quad + \frac{1}{2} m_2 g l_2 C_{12}. \tag{4.12}
 \end{aligned}$$

Equations (4.11) and (4.12) can also be written in a matrix form.

4.3 EFFECTIVE MOMENTS OF INERTIA

To simplify the equations of motion, Equations (4.5) and (4.6) or Equations (4.11) and (4.12) can be rewritten in symbolic form as follows:

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} D_{ii} & D_{ij} \\ D_{ji} & D_{jj} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_i \\ \ddot{\theta}_j \end{bmatrix} + \begin{bmatrix} D_{iii} & D_{ijj} \\ D_{jii} & D_{jjj} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \end{bmatrix} + \begin{bmatrix} D_{iij} & D_{iji} \\ D_{iji} & D_{jjj} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \dot{\theta}_2 \\ \dot{\theta}_2 \dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} D_i \\ D_j \end{bmatrix}. \quad (4.13)$$

In this equation, which is written for a two-degree-of-freedom system, a coefficient in the form of D_{ii} is known as effective inertia at joint i , such that an acceleration at joint i causes a torque at joint i equal to $D_{ii}\ddot{\theta}_i$, whereas a coefficient in the form D_{ij} is known as 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 $D_{ij}\ddot{\theta}_i$ or $D_{ji}\ddot{\theta}_j$. $D_{iij}\dot{\theta}_1^2$ terms represent centripetal forces acting at joint i due to a velocity at joint j . All terms with $\dot{\theta}_1\dot{\theta}_2$ represent Coriolis accelerations, and when multiplied by corresponding inertias, they will represent Coriolis forces. The remaining terms in the form D_i represent gravity forces at joint i .

4.4 DYNAMIC EQUATIONS FOR MULTIPLE-DEGREE-OF-FREEDOM ROBOTS

As you can see, the dynamic equations for a two-degree-of-freedom system is much more complicated than a one-degree-of-freedom system. Similarly, these equations for a multiple-degree-of-freedom robot are very long and complicated, but can be found by calculating the kinetic and potential energies of the links and the joints, by defining the Lagrangian, and by differentiating the Lagrangian equation with respect to the joint variables. The next section presents a summary of this procedure. For more information, please see [1,2,3,4,5,6,7].

4.4.1 Kinetic Energy

As you may remember from your dynamics course [8], the kinetic energy of a rigid body with motion in three dimensions is (Figure 4.7(a))

$$K = \frac{1}{2} m \bar{V}^2 + \frac{1}{2} \bar{\omega} \cdot \bar{h}_G, \quad (4.14)$$

where \bar{h}_G is the angular momentum of the body about G .

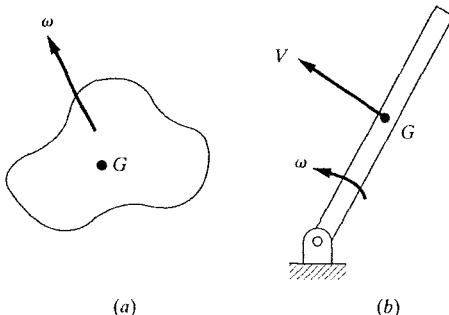


Figure 4.7 A rigid body in three-dimensional motion and in plane motion.

The kinetic energy of a rigid body in planar motion (Figure 4.7(b)) simplifies to

$$K = \frac{1}{2} m\bar{V}^2 + \frac{1}{2} \bar{I}\omega^2. \quad (4.15)$$

Thus, we will need to derive expressions for velocity of a point along a rigid body (e.g., the center of mass G), as well as the moments of inertia.

The velocity of a point along 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 will 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

$${}^R T_H = {}^R T_1 {}^1 T_2 {}^2 T_3 \dots {}^{n-1} T_n = A_1 A_2 A_3 \dots A_n. \quad (2.54)$$

For a six-axis robot, this equation can be written as

$${}^0 T_6 = {}^0 T_1 {}^1 T_2 {}^2 T_3 \dots {}^5 T_6 = A_1 A_2 A_3 \dots A_6. \quad (4.16)$$

Referring to Equation (2.52), we see that the derivative of an A_i matrix for a revolute joint with respect to its joint variable θ_i is

$$\begin{aligned} \frac{\partial A_i}{\partial \theta_i} &= \frac{\partial}{\partial \theta_i} \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} -S\theta_i & -C\theta_i C\alpha_i & C\theta_i S\alpha_i & -a_i S\theta_i \\ C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \quad (4.17)$$

However, this matrix can be broken into a constant matrix Q_i and the A_i matrix such that

$$\begin{bmatrix} -S\theta_i & -C\theta_i C\alpha_i & C\theta_i S\alpha_i & -a_i S\theta_i \\ C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (4.18)$$

or

$$\frac{\partial A_i}{\partial \theta_i} = Q_i A_i \quad (4.19)$$

Similarly, the derivative of an A_i matrix for a prismatic joint with respect to its joint variable is

$$\frac{\partial A_i}{\partial d_i} = \frac{\partial}{\partial d_i} \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad (4.20)$$

which, as before, can be broken into a constant matrix Q_i and the A_i matrix such that

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (4.21)$$

or

$$\frac{\partial A_i}{\partial \theta_i} = Q_i A_i \quad (4.22)$$

In both Equations (4.19) and (4.22), the Q_i matrices are always constant, as shown, and can be summarized as follows:

$$Q_i \text{ (revolute)} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad Q_i \text{ (prismatic)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (4.23)$$

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 Equation (4.16) with multiple joint variables (θ 's and d 's), differentiated with respect to only one variable q_j gives

$$U_{ij} = \frac{\partial {}^0T_i}{\partial q_j} = \frac{\partial (A_1 A_2 \dots A_j \dots A_i)}{\partial q_j} = A_1 A_2 \dots Q_j A_j \dots A_i, \quad j \leq i. \quad (4.24)$$

Please 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 be formulated similarly from

$$U_{ijk} = \partial U_{ij} / \partial q_k \quad (4.25)$$

Example 4.5

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. Thus,

$${}^0T_5 = A_1 A_2 A_3 A_4 A_5,$$

$$U_{52} = \frac{\partial {}^0T_5}{\partial \theta_2} = A_1 Q_2 A_2 A_3 A_4 A_5,$$

$$U_{53} = \frac{\partial^0 T_5}{\partial d_3} = A_1 A_2 Q_3 A_3 A_4 A_5,$$

where Q_2 and Q_3 are as defined in Equation (4.23).

Example 4.6

Find an expression for U_{635} of the Stanford Arm.

Solution

$${}^0 T_6 = A_1 A_2 A_3 A_4 A_5 A_6,$$

$$U_{63} = \frac{\partial^0 T_6}{\partial d_3} = A_1 A_2 Q_3 A_3 A_4 A_5 A_6,$$

$$U_{635} = \frac{\partial U_{63}}{\partial q_5} = A_1 A_2 Q_3 A_3 A_4 Q_5 A_5 A_6.$$

4.4.1 Continued

We will now continue with the derivation of the velocity term for a point on a link of a robot. Using r_i to represent a point on any link i of the robot relative to frame i , we can express the position of the point by premultiplying the vector with the transformation matrix representing its frame:

$$p_i = {}^R T_i r_i = {}^0 T_i r_i. \quad (4.26)$$

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 Equation (4.26) with respect to all the joint variables q_j yields the velocity of the point:

$$V_i = \frac{d}{dt} ({}^0 T_i r_i) = \sum_{j=1}^i \left(\frac{\partial ({}^0 T_i)}{\partial q_j} \frac{dq_j}{dt} \right) r_i = \sum_{j=1}^i \left(U_{ij} \frac{dq_j}{dt} \right) \cdot r_i. \quad (4.27)$$

The kinetic energy of an element of mass m_i on a link is

$$dK_i = \frac{1}{2} (\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2) dm_i. \quad (4.28)$$

Since V_i has three components of $\dot{x}_i, \dot{y}_i, \dot{z}_i$, it can be written as a matrix, and thus

$$V_i V_i^T = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{z}_i \end{bmatrix} \begin{bmatrix} \dot{x}_i & \dot{x}_i \dot{y}_i & \dot{x}_i \dot{z}_i \\ \dot{y}_i \dot{x}_i & \dot{y}_i^2 & \dot{y}_i \dot{z}_i \\ \dot{z}_i \dot{x}_i & \dot{z}_i \dot{y}_i & \dot{z}_i^2 \end{bmatrix}$$

and

$$\text{Trace}(V_i V_i^T) = \text{Trace} \begin{bmatrix} \dot{x}_i^2 & \dot{x}_i \dot{y}_i & \dot{x}_i \dot{z}_i \\ \dot{y}_i \dot{x}_i & \dot{y}_i^2 & \dot{y}_i \dot{z}_i \\ \dot{z}_i \dot{x}_i & \dot{z}_i \dot{y}_i & \dot{z}_i^2 \end{bmatrix} = \dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2 \quad (4.29)$$

Combining Equations (4.27), (4.28), and (4.29) yields the following equation for the kinetic energy of the element:

$$dK_i = \frac{1}{2} \text{Trace} \left[\left(\sum_{p=1}^i \left(U_{ip} \frac{dq_p}{dt} \right) \cdot r_i \right) \left(\sum_{r=1}^i \left(U_{ir} \frac{dq_r}{dt} \right) \cdot r_i \right)^T \right] dm_i \quad (4.30)$$

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 terms yields the total kinetic energy:

$$K_i = \int dK_i = \frac{1}{2} \text{Trace} \left[\sum_{p=1}^i \sum_{r=1}^i U_{ip} (\int r_i r_i^T dm_i) U_{ir}^T \dot{q}_p \dot{q}_r \right]. \quad (4.31)$$

The Pseudo Inertia Matrix, representing the $\int r_i r_i^T dm_i$ terms, can be written as

$$J_i = \begin{bmatrix} (-I_{xx} + I_{yy} + I_{zz})/2 & I_{xy} & I_{xz} & m_i \bar{x}_i \\ I_{xy} & (I_{xx} - I_{yy} + I_{zz})/2 & I_{yz} & m_i \bar{y}_i \\ I_{xz} & I_{yz} & (I_{xx} + I_{yy} - I_{zz})/2 & m_i \bar{z}_i \\ m_i \bar{x}_i & m_i \bar{y}_i & m_i \bar{z}_i & m_i \end{bmatrix}. \quad (4.32)$$

Since this matrix is independent of joint angles and velocities, it must be evaluated only once. Substituting Equation (4.32) into Equation (4.31) gives the final form for kinetic energy of the robot manipulator:

$$K = \frac{1}{2} \sum_{i=1}^n \sum_{p=1}^i \sum_{r=1}^i \text{Trace} (U_{ip} J_i U_{ir}^T) \dot{q}_p \dot{q}_r. \quad (4.33)$$

The kinetic energy of the actuators can also be added to this equation. Assuming that each actuator has an inertia of $I_{i(\text{act})}$, the kinetic energy of the actuator is $1/2 I_{i(\text{act})} \dot{q}_i^2$, and the total kinetic energy of the robot is

$$K = \frac{1}{2} \sum_{i=1}^n \sum_{p=1}^i \sum_{r=1}^i \text{Trace} (U_{ip} J_i U_{ir}^T) \dot{q}_p \dot{q}_r + 1/2 \sum_{i=1}^n I_{i(\text{act})} \dot{q}_i^2. \quad (4.34)$$

4.4.2 Potential Energy

The potential energy of the system is the sum of the potential energies of each link and can be written as

$$P = \sum_{i=1}^n P_i = \sum_{i=1}^n [-m_i g^T \cdot (^0 T_i \bar{r}_i)], \quad (4.35)$$

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 thus g^T , which is a (1×4) matrix, multiplied by the position vector $(^0 T_i \bar{r}_i)$, which is a (4×1) matrix, yields a single scalar quantity. Please also notice that the values in the gravity matrix are dependent on the orientation of the reference frame.

4.4.3 The Lagrangian

The Lagrangian is then

$$\begin{aligned} L = K - P = & \frac{1}{2} \sum_{i=1}^n \sum_{p=1}^i \sum_{r=1}^i \text{Trace}(U_{ip} J_i U_{ir}^T) \dot{q}_p \dot{q}_r \\ & + 1/2 \sum_{i=1}^n I_{i(\text{act})} \dot{q}_i^2 - \sum_{i=1}^n [-m_i g^T \cdot {}^0 T_i \bar{r}_i]. \end{aligned} \quad (4.36)$$

4.4.4 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:

$$T_i = \sum_{j=1}^n D_{ij} \ddot{q}_j + I_{i(\text{act})} \dot{q}_i + \sum_{j=1}^n \sum_{k=1}^n D_{ijk} \dot{q}_j \dot{q}_k + D_i, \quad (4.37)$$

where

$$D_{ij} = \sum_{p=\max(i,j)}^n \text{Trace}(U_{pj} J_p U_{pi}^T) \quad (4.38)$$

and

$$D_{ijk} = \sum_{p=\max(i,j,k)}^n \text{Trace}(U_{pj} J_p U_{pi}^T), \quad (4.39)$$

and

$$D_i = \sum_{p=i}^n [-m_p g^T U_{pi} \bar{r}_p]. \quad (4.40)$$

In Equation (4.37), the first part is the angular acceleration-inertia terms, the second part is the actuator inertia term, the third part is the Coriolis and centrifugal terms, and the last part is the gravity term. This equation can be expanded for a six-axis revolute robot as follows:

$$\begin{aligned} T_i = & D_{i1} \ddot{\theta}_1 + D_{i2} \ddot{\theta}_2 + D_{i3} \ddot{\theta}_3 + D_{i4} \ddot{\theta}_4 + D_{i5} \ddot{\theta}_5 + D_{i6} \ddot{\theta}_6 + I_{i(\text{act})} \ddot{\theta}_i \\ & + D_{i11} \dot{\theta}_1^2 + D_{i22} \dot{\theta}_2^2 + D_{i33} \dot{\theta}_3^2 + D_{i44} \dot{\theta}_4^2 + D_{i55} \dot{\theta}_5^2 + D_{i66} \dot{\theta}_6^2 \\ & + D_{i12} \dot{\theta}_1 \dot{\theta}_2 + D_{i13} \dot{\theta}_1 \dot{\theta}_3 + D_{i14} \dot{\theta}_1 \dot{\theta}_4 + D_{i15} \dot{\theta}_1 \dot{\theta}_5 + D_{i16} \dot{\theta}_1 \dot{\theta}_6 \\ & + D_{i23} \dot{\theta}_2 \dot{\theta}_3 + D_{i24} \dot{\theta}_2 \dot{\theta}_4 + D_{i25} \dot{\theta}_2 \dot{\theta}_5 + D_{i26} \dot{\theta}_2 \dot{\theta}_6 \\ & + D_{i34} \dot{\theta}_3 \dot{\theta}_4 + D_{i35} \dot{\theta}_3 \dot{\theta}_5 + D_{i36} \dot{\theta}_3 \dot{\theta}_6 \\ & + D_{i45} \dot{\theta}_4 \dot{\theta}_5 + D_{i46} \dot{\theta}_4 \dot{\theta}_6 \\ & + D_{i56} \dot{\theta}_5 \dot{\theta}_6 + D_{i53} \dot{\theta}_5 \dot{\theta}_3 + D_{i54} \dot{\theta}_5 \dot{\theta}_4 + D_{i56} \dot{\theta}_5 \dot{\theta}_6 \\ & + D_{i61} \dot{\theta}_6 \dot{\theta}_1 + D_{i62} \dot{\theta}_6 \dot{\theta}_2 + D_{i63} \dot{\theta}_6 \dot{\theta}_3 + D_{i64} \dot{\theta}_6 \dot{\theta}_4 + D_{i65} \dot{\theta}_6 \dot{\theta}_5 + D_i. \end{aligned} \quad (4.41)$$

Notice that in Equation (4.41) there are two terms with $\dot{\theta}_1\dot{\theta}_2$. The two coefficients are D_{512} and D_{521} . To see what these terms look like, let's calculate them for $i = 5$. From Equation (4.39), for D_{512} , we will 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

$$\begin{aligned} D_{512} &= \text{Trace}(U_{512}J_5U_{55}^T) + \text{Trace}(U_{612}J_6U_{65}^T) \\ D_{521} &= \text{Trace}(U_{521}J_5U_{55}^T) + \text{Trace}(U_{621}J_6U_{65}^T) \end{aligned} \quad (4.42)$$

and from Equation (4.24) we have

$$\begin{aligned} U_{51} &= \frac{\partial A_1 A_2 A_3 A_4 A_5}{\partial \theta_1} = Q_1 A_1 A_2 A_3 A_4 A_5 \\ \rightarrow U_{512} &= U_{(51)2} = \frac{\partial(Q_1 A_1 A_2 A_3 A_4 A_5)}{\partial \theta_2} = Q_1 A_1 Q_2 A_2 A_3 A_4 A_5, \\ U_{52} &= \frac{\partial A_1 A_2 A_3 A_4 A_5}{\partial \theta_2} = A_1 Q_2 A_2 A_3 A_4 A_5 \\ \rightarrow U_{521} &= U_{(52)1} = \frac{\partial(A_1 Q_2 A_2 A_3 A_4 A_5)}{\partial \theta_1} = Q_1 A_1 Q_2 A_2 A_3 A_4 A_5, \\ U_{61} &= \frac{\partial A_1 A_2 A_3 A_4 A_5 A_6}{\partial \theta_1} = Q_1 A_1 A_2 A_3 A_4 A_5 A_6 \\ \rightarrow U_{612} &= U_{(61)2} = \frac{\partial(Q_1 A_1 A_2 A_3 A_4 A_5 A_6)}{\partial \theta_2} = Q_1 A_1 Q_2 A_2 A_3 A_4 A_5 A_6, \\ U_{62} &= \frac{\partial A_1 A_2 A_3 A_4 A_5 A_6}{\partial \theta_2} = A_1 Q_2 A_2 A_3 A_4 A_5 A_6 \\ \rightarrow U_{621} &= U_{(62)1} = \frac{\partial(A_1 Q_2 A_2 A_3 A_4 A_5 A_6)}{\partial \theta_1} = Q_1 A_1 Q_2 A_2 A_3 A_4 A_5 A_6. \end{aligned} \quad (4.43)$$

Please 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 Equation (4.43) into (4.42) shows that $D_{512} = D_{521}$. Clearly, the summation of the two similar terms yields the corresponding Coriolis acceleration term for $\dot{\theta}_1\dot{\theta}_2$. This is true for all similar coefficients in Equation (4.41). Thus, we can simplify this equation for all joints as follows:

$$\begin{aligned} T_1 &= D_{11}\ddot{\theta}_1 + D_{12}\ddot{\theta}_2 + D_{13}\ddot{\theta}_3 + D_{14}\ddot{\theta}_4 + D_{15}\ddot{\theta}_5 + D_{16}\ddot{\theta}_6 + I_{1(\text{act})}\ddot{\theta}_1 \\ &\quad + D_{111}\dot{\theta}_1^2 + D_{122}\dot{\theta}_2^2 + D_{133}\dot{\theta}_3^2 + D_{144}\dot{\theta}_4^2 + D_{155}\dot{\theta}_5^2 + D_{166}\dot{\theta}_6^2 \\ &\quad + 2D_{112}\dot{\theta}_1\dot{\theta}_2 + 2D_{113}\dot{\theta}_1\dot{\theta}_3 + 2D_{114}\dot{\theta}_1\dot{\theta}_4 + 2D_{115}\dot{\theta}_1\dot{\theta}_5 + 2D_{116}\dot{\theta}_1\dot{\theta}_6 \\ &\quad + 2D_{123}\dot{\theta}_2\dot{\theta}_3 + 2D_{124}\dot{\theta}_2\dot{\theta}_4 + 2D_{125}\dot{\theta}_2\dot{\theta}_5 + 2D_{126}\dot{\theta}_2\dot{\theta}_6 + 2D_{134}\dot{\theta}_3\dot{\theta}_4 \\ &\quad + 2D_{135}\dot{\theta}_3\dot{\theta}_5 + 2D_{136}\dot{\theta}_3\dot{\theta}_6 + 2D_{145}\dot{\theta}_4\dot{\theta}_5 + 2D_{146}\dot{\theta}_4\dot{\theta}_6 + 2D_{156}\dot{\theta}_5\dot{\theta}_6 + D_1, \end{aligned} \quad (4.44)$$

$$\begin{aligned}
T_2 = & D_{21}\ddot{\theta}_1 + D_{22}\ddot{\theta}_2 + D_{23}\ddot{\theta}_3 + D_{24}\ddot{\theta}_4 + D_{25}\ddot{\theta}_5 + D_{26}\ddot{\theta}_6 + I_{2(\text{act})}\ddot{\theta}_2 \\
& + D_{211}\dot{\theta}_1^2 + D_{222}\dot{\theta}_2^2 + D_{233}\dot{\theta}_3^2 + D_{244}\dot{\theta}_4^2 + D_{255}\dot{\theta}_5^2 + D_{266}\dot{\theta}_6^2 \\
& + 2D_{212}\dot{\theta}_1\dot{\theta}_2 + 2D_{213}\dot{\theta}_1\dot{\theta}_3 + 2D_{214}\dot{\theta}_1\dot{\theta}_4 + 2D_{215}\dot{\theta}_1\dot{\theta}_5 + 2D_{216}\dot{\theta}_1\dot{\theta}_6 \\
& + 2D_{223}\dot{\theta}_2\dot{\theta}_3 + 2D_{224}\dot{\theta}_2\dot{\theta}_4 + 2D_{225}\dot{\theta}_2\dot{\theta}_5 + 2D_{226}\dot{\theta}_2\dot{\theta}_6 + 2D_{234}\dot{\theta}_3\dot{\theta}_4 \\
& + 2D_{235}\dot{\theta}_3\dot{\theta}_5 + 2D_{236}\dot{\theta}_3\dot{\theta}_6 + 2D_{245}\dot{\theta}_4\dot{\theta}_5 + 2D_{246}\dot{\theta}_4\dot{\theta}_6 + 2D_{256}\dot{\theta}_5\dot{\theta}_6 + D_2,
\end{aligned} \tag{4.45}$$

$$\begin{aligned}
T_3 = & D_{31}\ddot{\theta}_1 + D_{32}\ddot{\theta}_2 + D_{33}\ddot{\theta}_3 + D_{34}\ddot{\theta}_4 + D_{35}\ddot{\theta}_5 + D_{36}\ddot{\theta}_6 + I_{3(\text{act})}\ddot{\theta}_3 \\
& + D_{311}\dot{\theta}_1^2 + D_{322}\dot{\theta}_2^2 + D_{333}\dot{\theta}_3^2 + D_{344}\dot{\theta}_4^2 + D_{355}\dot{\theta}_5^2 + D_{366}\dot{\theta}_6^2 \\
& + 2D_{312}\dot{\theta}_1\dot{\theta}_2 + 2D_{313}\dot{\theta}_1\dot{\theta}_3 + 2D_{314}\dot{\theta}_1\dot{\theta}_4 + 2D_{315}\dot{\theta}_1\dot{\theta}_5 + 2D_{316}\dot{\theta}_1\dot{\theta}_6 \\
& + 2D_{323}\dot{\theta}_2\dot{\theta}_3 + 2D_{324}\dot{\theta}_2\dot{\theta}_4 + 2D_{325}\dot{\theta}_2\dot{\theta}_5 + 2D_{326}\dot{\theta}_2\dot{\theta}_6 + 2D_{334}\dot{\theta}_3\dot{\theta}_4 \\
& + 2D_{335}\dot{\theta}_3\dot{\theta}_5 + 2D_{336}\dot{\theta}_3\dot{\theta}_6 + 2D_{345}\dot{\theta}_4\dot{\theta}_5 + 2D_{346}\dot{\theta}_4\dot{\theta}_6 + 2D_{356}\dot{\theta}_5\dot{\theta}_6 + D_3,
\end{aligned} \tag{4.46}$$

$$\begin{aligned}
T_4 = & D_{41}\ddot{\theta}_1 + D_{42}\ddot{\theta}_2 + D_{43}\ddot{\theta}_3 + D_{44}\ddot{\theta}_4 + D_{45}\ddot{\theta}_5 + D_{46}\ddot{\theta}_6 + I_{4(\text{act})}\ddot{\theta}_4 \\
& + D_{411}\dot{\theta}_1^2 + D_{422}\dot{\theta}_2^2 + D_{433}\dot{\theta}_3^2 + D_{444}\dot{\theta}_4^2 + D_{455}\dot{\theta}_5^2 + D_{466}\dot{\theta}_6^2 \\
& + 2D_{412}\dot{\theta}_1\dot{\theta}_2 + 2D_{413}\dot{\theta}_1\dot{\theta}_3 + 2D_{414}\dot{\theta}_1\dot{\theta}_4 + 2D_{415}\dot{\theta}_1\dot{\theta}_5 + 2D_{416}\dot{\theta}_1\dot{\theta}_6 \\
& + 2D_{423}\dot{\theta}_2\dot{\theta}_3 + 2D_{424}\dot{\theta}_2\dot{\theta}_4 + 2D_{425}\dot{\theta}_2\dot{\theta}_5 + 2D_{426}\dot{\theta}_2\dot{\theta}_6 + 2D_{434}\dot{\theta}_3\dot{\theta}_4 \\
& + 2D_{435}\dot{\theta}_3\dot{\theta}_5 + 2D_{436}\dot{\theta}_3\dot{\theta}_6 + 2D_{445}\dot{\theta}_4\dot{\theta}_5 + 2D_{446}\dot{\theta}_4\dot{\theta}_6 + 2D_{456}\dot{\theta}_5\dot{\theta}_6 + D_4,
\end{aligned} \tag{4.47}$$

$$\begin{aligned}
T_5 = & D_{51}\ddot{\theta}_1 + D_{52}\ddot{\theta}_2 + D_{53}\ddot{\theta}_3 + D_{54}\ddot{\theta}_4 + D_{55}\ddot{\theta}_5 + D_{56}\ddot{\theta}_6 + I_{5(\text{act})}\ddot{\theta}_5 \\
& + D_{511}\dot{\theta}_1^2 + D_{522}\dot{\theta}_2^2 + D_{533}\dot{\theta}_3^2 + D_{544}\dot{\theta}_4^2 + D_{555}\dot{\theta}_5^2 + D_{566}\dot{\theta}_6^2 \\
& + 2D_{512}\dot{\theta}_1\dot{\theta}_2 + 2D_{513}\dot{\theta}_1\dot{\theta}_3 + 2D_{514}\dot{\theta}_1\dot{\theta}_4 + 2D_{515}\dot{\theta}_1\dot{\theta}_5 + 2D_{516}\dot{\theta}_1\dot{\theta}_6 \\
& + 2D_{523}\dot{\theta}_2\dot{\theta}_3 + 2D_{524}\dot{\theta}_2\dot{\theta}_4 + 2D_{525}\dot{\theta}_2\dot{\theta}_5 + 2D_{526}\dot{\theta}_2\dot{\theta}_6 + 2D_{534}\dot{\theta}_3\dot{\theta}_4 \\
& + 2D_{535}\dot{\theta}_3\dot{\theta}_5 + 2D_{536}\dot{\theta}_3\dot{\theta}_6 + 2D_{545}\dot{\theta}_4\dot{\theta}_5 + 2D_{546}\dot{\theta}_4\dot{\theta}_6 + 2D_{556}\dot{\theta}_5\dot{\theta}_6 + D_5,
\end{aligned} \tag{4.48}$$

$$\begin{aligned}
T_6 = & D_{61}\ddot{\theta}_1 + D_{62}\ddot{\theta}_2 + D_{63}\ddot{\theta}_3 + D_{64}\ddot{\theta}_4 + D_{65}\ddot{\theta}_5 + D_{66}\ddot{\theta}_6 + I_{6(\text{act})}\ddot{\theta}_6 \\
& + D_{611}\dot{\theta}_1^2 + D_{622}\dot{\theta}_2^2 + D_{633}\dot{\theta}_3^2 + D_{644}\dot{\theta}_4^2 + D_{655}\dot{\theta}_5^2 + D_{666}\dot{\theta}_6^2 \\
& + 2D_{612}\dot{\theta}_1\dot{\theta}_2 + 2D_{613}\dot{\theta}_1\dot{\theta}_3 + 2D_{614}\dot{\theta}_1\dot{\theta}_4 + 2D_{615}\dot{\theta}_1\dot{\theta}_5 + 2D_{616}\dot{\theta}_1\dot{\theta}_6 \\
& + 2D_{623}\dot{\theta}_2\dot{\theta}_3 + 2D_{624}\dot{\theta}_2\dot{\theta}_4 + 2D_{625}\dot{\theta}_2\dot{\theta}_5 + 2D_{626}\dot{\theta}_2\dot{\theta}_6 + 2D_{634}\dot{\theta}_3\dot{\theta}_4 \\
& + 2D_{635}\dot{\theta}_3\dot{\theta}_5 + 2D_{636}\dot{\theta}_3\dot{\theta}_6 + 2D_{645}\dot{\theta}_4\dot{\theta}_5 + 2D_{646}\dot{\theta}_4\dot{\theta}_6 + 2D_{656}\dot{\theta}_5\dot{\theta}_6 + D_6.
\end{aligned} \tag{4.49}$$

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. 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 centrifugal and Coriolis accelerations may become negligible. In general, using these equations, one can properly design and control a robot.

Example 4.7

Using the aforementioned equations, derive the equations of motion for the two-degree-of-freedom robot arm of Example 4.4. The two links are assumed to be of equal length.

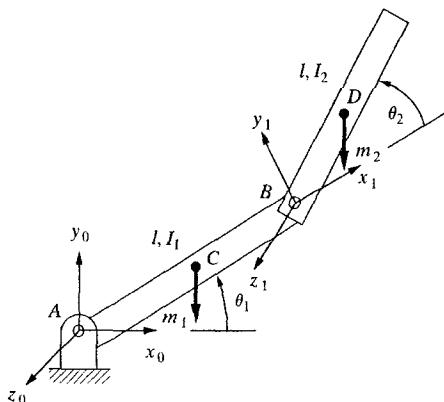


Figure 4.8 The two-degree-of-freedom robot arm of Example 4.4.

Solution To use the equations of motion for the robot, we will first write the A matrices for the two links; then we will develop the D_{ij} , D_{ijk} , and D_i terms for the robot. Finally, we substitute the results into Equations (4.44) and (4.45) to get the final equations of motion. The joint and link parameters of the robot are $d_1 = 0$, $d_2 = 0$, $a_1 = l$, $a_2 = l$, $\alpha_1 = 0$, $\alpha_2 = 0$.

$$A_1 = \begin{bmatrix} C_1 & -S_1 & 0 & IC_1 \\ S_1 & C_1 & 0 & IS_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & IC_2 \\ S_2 & C_2 & 0 & IS_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0T_2 = A_1 A_2 = \begin{bmatrix} C_{12} & -S_{12} & 0 & l(C_{12} + C_1) \\ S_{12} & C_{12} & 0 & l(S_{12} + S_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\text{From Equation (4.23), } Q \text{ (revolute)} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

$$\text{From Equation (4.24), we have } U_{ij} = \frac{\partial {}^0T_i}{\partial q_j} = \frac{\partial (A_1 A_2 \dots A_j \dots A_i)}{\partial q_j} = A_1 A_2 \dots Q_j A_j \dots A_i. \text{ Thus,}$$

$$U_{11} = QA_1 = \begin{bmatrix} -S_1 & -C_1 & 0 & -lS_1 \\ C_1 & -S_1 & 0 & lC_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\rightarrow U_{111} = \frac{\partial(QA_1)}{\partial\theta_1} = QQA_1 \text{ and } U_{112} = \frac{\partial(QA_1)}{\partial\theta_2} = 0,$$

$$U_{21} = QA_1A_2 = \begin{bmatrix} -S_{12} & -C_{12} & 0 & -l(S_{12} + S_1) \\ C_{12} & -S_{12} & 0 & l(C_{12} + C_1) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\rightarrow U_{211} = \frac{\partial(QA_1A_2)}{\partial\theta_1} = QQA_1A_2 \text{ and } U_{212} = \frac{\partial(QA_1A_2)}{\partial\theta_2} = QA_1QA_2,$$

$$U_{22} = A_1QA_2 = \begin{bmatrix} -S_{12} & -C_{12} & 0 & -lS_{12} \\ C_{12} & -S_{12} & 0 & lC_{12} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\rightarrow U_{221} = \frac{\partial(A_1QA_2)}{\partial\theta_1} = QA_1QA_2 \text{ and } U_{222} = \frac{\partial(A_1QA_2)}{\partial\theta_2} = A_1QQA_2,$$

$$U_{12} = \frac{\partial A_1}{\partial\theta_2} = 0.$$

From Equation (4.32), assuming that all products of inertia are zero, we have

$$J_1 = \begin{bmatrix} \frac{1}{3}m_1l^2 & 0 & 0 & -\frac{1}{2}m_1l \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{2}m_1l & 0 & 0 & m_1 \end{bmatrix}, \quad J_2 = \begin{bmatrix} \frac{1}{3}m_2l^2 & 0 & 0 & -\frac{1}{2}m_2l \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{2}m_2l & 0 & 0 & m_2 \end{bmatrix}.$$

From Equations (4.44) and (4.45), for a two-degree-of-freedom robot, we get

$$T_1 = D_{11}\ddot{\theta}_1 + D_{12}\ddot{\theta}_2 + D_{111}\dot{\theta}_1^2 + D_{122}\dot{\theta}_2^2 + 2D_{112}\dot{\theta}_1\dot{\theta}_2 + D_1 + I_{1(\text{act})}\ddot{\theta}_1, \quad (4.50)$$

$$T_2 = D_{21}\ddot{\theta}_1 + D_{22}\ddot{\theta}_2 + D_{211}\dot{\theta}_1^2 + D_{222}\dot{\theta}_2^2 + 2D_{212}\dot{\theta}_1\dot{\theta}_2 + D_2 + I_{2(\text{act})}\ddot{\theta}_2. \quad (4.51)$$

From Equations (4.38), (4.39), and (4.40), we have

$$D_{11} = \text{Trace}(U_{11}J_1U_{11}^T) + \text{Trace}(U_{21}J_2U_{21}^T), \quad \text{for } i = 1, j = 1, p = 1, 2,$$

$$D_{12} = \text{Trace}(U_{22}J_2U_{21}^T), \quad \text{for } i = 1, j = 2, p = 2,$$

$$D_{21} = \text{Trace}(U_{21}J_2U_{22}^T), \quad \text{for } i = 2, j = 1, p = 2,$$

$$D_{22} = \text{Trace}(U_{22}J_2U_{22}^T), \quad \text{for } i = 2, j = 2, p = 2,$$

$$D_{111} = \text{Trace}(U_{11}J_1U_{11}^T) + \text{Trace}(U_{21}J_2U_{21}^T), \quad \text{for } i = 1, j = 1, k = 1, p = 1, 2,$$

$$\begin{aligned}
D_{122} &= \text{Trace}(U_{222}J_2U_{21}^T), & \text{for } i = 1, j = 2, k = 2, p = 2, \\
D_{112} &= \text{Trace}(U_{212}J_2U_{21}^T), & \text{for } i = 1, j = 1, k = 2, p = 2, \\
D_{211} &= \text{Trace}(U_{211}J_2U_{22}^T), & \text{for } i = 2, j = 1, k = 1, p = 2, \\
D_{222} &= \text{Trace}(U_{222}J_2U_{22}^T), & \text{for } i = 2, j = 2, k = 2, p = 2, \\
D_{212} &= \text{Trace}(U_{212}J_2U_{22}^T), & \text{for } i = 2, j = 1, k = 2, p = 2, \\
D_1 &= -m_1g^T U_{11}\bar{r}_1 - m_2g^T U_{21}\bar{r}_2, & \text{for } i = 1, p = 1, 2, \\
D_2 &= -m_1g^T U_{12}\bar{r}_1 - m_2g^T U_{22}\bar{r}_2, & \text{for } i = 2, p = 1, 2.
\end{aligned}$$

Although forbiddingly long, even for a two-degree-of-freedom robot, substituting all given matrices into these equations yields

$$\begin{aligned}
D_{11} &= \frac{1}{3}m_1l^2 + \frac{4}{3}m_2l^2 + m_2l^2C_2, \\
D_{12} &= D_{21} = \frac{1}{3}m_2l^2 + \frac{1}{2}m_2l^2C_2, \\
D_{22} &= \frac{1}{3}m_2l^2, \\
D_{111} &= 0, & D_{112} = D_{121} = -\frac{1}{2}m_2l^2S_2, \\
D_{122} &= -\frac{1}{2}m_2l^2S_2 & D_{211} = \frac{1}{2}m_2l^2S_2, \\
D_{212} &= 0, & D_{221} = 0, \quad D_{222} = 0,
\end{aligned}$$

and for $g^T = [0 \ -g \ 0 \ 0]$ and $r_1^T = r_2^T = -\frac{l}{2} \ 0 \ 0 \ 1$, 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 Equations (4.50) and (4.51) gives the final equations of motion, which, except for the actuator inertia terms, are the same as Equations (4.11) and (4.12):

$$\begin{aligned}
T_1 &= \left(\frac{1}{3}m_1l^2 + \frac{4}{3}m_2l^2 + m_2l^2C_2 \right) \ddot{\theta}_1 + \left(\frac{1}{3}m_2l^2 + \frac{1}{2}m_2l^2C_2 \right) \ddot{\theta}_2 \\
&\quad + \left(\frac{1}{2}m_2l^2S_2 \right) \dot{\theta}_2^2 + (m_2l^2S_2)\dot{\theta}_1\dot{\theta}_2 + \frac{1}{2}m_1glC_1 + \frac{1}{2}m_2glC_{12} + m_2glC_1 + I_{1(\text{act})}\ddot{\theta}_1, \\
T_2 &= \left(\frac{1}{3}m_2l^2 + \frac{1}{2}m_2l^2C_2 \right) \ddot{\theta}_1 + \left(\frac{1}{3}m_2l^2 \right) \ddot{\theta}_2 + \left(\frac{1}{2}m_2l^2S_2 \right) \dot{\theta}_1^2 \\
&\quad + \frac{1}{2}m_2glC_{12} + I_{2(\text{act})}\ddot{\theta}_1.
\end{aligned}$$

4.5 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. So long as the surface is flat and as long as the robot is following the line on the flat surface, the groove will be uniform. However, if the surface has a slight unknown curvature in it, since the robot is following a given path, it will either cut deeper into the surface, or not cut deep enough. Alternatively, suppose that 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 that 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.

To relate the joint forces and torques to forces and moments generated at the hand frame of the robot [1,9,10], we will define

$$[{}^H F] = [f_x \ f_y \ f_z \ m_x \ m_y \ m_z]^T, \quad (4.52)$$

where f_x, f_y, f_z are the forces along the $\bar{x}, \bar{y}, \bar{z}$ axes of the hand frame, and m_x, m_y, m_z are the moments about the $\bar{x}, \bar{y}, \bar{z}$ axes of the hand frame. Similarly, we define

$$[{}^H D] = [dx \ dy \ dz \ \delta x \ \delta y \ \delta z]^T, \quad (4.53)$$

which expresses displacements and rotations about the $\bar{x}, \bar{y}, \bar{z}$ axes of the hand frame. We can also define similar entities for the joints as

$$[T] = [T_1 \ T_2 \ T_3 \ T_4 \ T_5 \ T_6]^T, \quad (4.54)$$

which are the torques (for revolute joints) and forces (for prismatic joints) at each joint, and

$$[D_\theta] = [d\theta_1 \ d\theta_2 \ d\theta_3 \ d\theta_4 \ d\theta_5 \ d\theta_6]^T, \quad (4.55)$$

which describes the differential movements at the joints, either an angle for 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 work at the hand frame, we get

$$\delta W = [{}^H F]^T [{}^H D] = [T]^T [D_\theta], \quad (4.56)$$

or that the forces times the displacements at the hand frame is equal to the torques (or forces) times the displacements at the joints. (Can you tell why the transpose of the force and torque matrices are used?). Substituting the values, we will get the following for the left-hand side of Equation (4.56):

$$\begin{bmatrix} f_x & f_y & f_z & m_x & m_y & m_z \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \\ \delta z \end{bmatrix} = f_x dx + f_y dy + \cdots + m_z \delta z. \quad (4.57)$$

However, from Equation (3.24), we have

$$[{}^T D] = [{}^T J] [D_\theta], \quad \text{or} \quad [{}^H D] = [{}^H J] [D_\theta].$$

Substituting this into Equation (4.56) gives:

$$[{}^H F]^T [{}^H J] [D_\theta] = [T]^T [D_\theta] \rightarrow [{}^H F]^T [{}^H J] = [T]^T. \quad (4.58)$$

Referring to Appendix A, this equation can be written as

$$[T] = [{}^H J]^T [{}^H F], \quad (4.59)$$

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 this later in Chapter 7.

Example 4.8

The numerical value of the Jacobian of a spherical-RPY robot (e.g., the Stanford Arm) is given next. 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 Equation (4.59), 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 can see, this means that for this particular configuration of the robot, with the specific dimensions it has, 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. As you notice, there is no need for the third joint, the prismatic joint, to exert any force, even though we desire a force at the hand frame. Can you visualize why?

Obviously, as the configuration of the robot changes, the Jacobian changes as well. Thus, for continued exertion of the same force and moment at the hand frame as the robot moves, the joint torques will have to change as well, requiring continuous calculation of joint torques by the controller.

4.6 TRANSFORMATION OF FORCES AND MOMENTS BETWEEN COORDINATE FRAMES

Suppose that two coordinate frames are attached to an object and that a force and a moment are acting on the object, and are described with respect to one of the coordinate frames. The same principle of virtual work can also be used here to find an equivalent force and moment with respect to the other coordinate frame such that they will have the same effect on the object. To do this, we will define F to be the forces and moments acting on the object and D to be the displacements caused by these forces and moments, also relative to the same reference frame:

$$[F]^T = [f_x, f_y, f_z, m_x, m_y, m_z], \quad (4.60)$$

$$[D]^T = [d_x, d_y, d_z, \delta_x, \delta_y, \delta_z]. \quad (4.61)$$

We will also define ${}^B F$ to be the forces and moments acting on the object relative to a coordinate frame B , and ${}^B D$ to be the displacements caused by these forces and moments, also relative to the coordinate frame B :

$${}^B F^T = [{}^B f_x, {}^B f_y, {}^B f_z, {}^B m_x, {}^B m_y, {}^B m_z], \quad (4.62)$$

$${}^B D^T = [{}^B d_x, {}^B d_y, {}^B d_z, {}^B \delta_x, {}^B \delta_y, {}^B \delta_z]. \quad (4.63)$$

Since the total virtual work performed on the object in either frame must be the same,

$$\delta W = [F]^T [D] = [{}^B F]^T [{}^B D]. \quad (4.64)$$

Paul [1] has shown that displacements relative to the two frames are related to each other by the following relationship:

$$[{}^B D] = [{}^R J] [D]. \quad (4.65)$$

Substituting Equation (4.65) into Equation (4.64) results in

$$[F]^T [D] = [{}^B F]^T [{}^B J] [D], \text{ or } [F]^T = [{}^B F]^T [{}^B J], \quad (4.66)$$

which can be rearranged as

$$[F] = [{}^B J]^T [{}^B F]. \quad (4.67)$$

Paul [1] has also shown that instead of calculating the Jacobian with respect to frame B , one can find the forces and moments with respect to frame B directly from the following equations:

$$\begin{aligned} {}^B f_x &= \bar{n} \cdot \bar{f}, \\ {}^B f_y &= \bar{o} \cdot \bar{f}, \\ {}^B f_z &= \bar{a} \cdot \bar{f}, \\ {}^B m_x &= \bar{n} \cdot [(\bar{f} \times \bar{p}) + \bar{m}], \\ {}^B m_y &= \bar{o} \cdot [(\bar{f} \times \bar{p}) + \bar{m}], \\ {}^B m_z &= \bar{a} \cdot [(\bar{f} \times \bar{p}) + \bar{m}]. \end{aligned} \quad (4.68)$$

Using these equations, we may find equivalent forces and moments in different frames that will have the same effect on an object.

Example 4.9

An object, attached to a frame B , is subjected to the following forces and moments relative to the reference frame:

$$F^T = [0, 10 \text{ lb}, 0, 0, 0, 20 \text{ lb} \cdot \text{in}],$$

$$B = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 5 \\ 1 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Find the equivalent forces and moments in frame B .

Solution From the given information, we have:

$$\bar{f} = [0, 10, 0], \quad \bar{m} = [0, 0, 20], \quad \bar{p} = [3, 5, 8].$$

$$\bar{n} = [0, 0, 1], \quad \bar{o} = [1, 0, 0], \quad \bar{a} = [0, 1, 0].$$

$$\bar{f} \times \bar{p} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ 0 & 10 & 0 \\ 3 & 5 & 8 \end{vmatrix} = \hat{i}(80) - \hat{j}(0) + \hat{k}(-30)$$

$$(\bar{f} \times \bar{p}) + \bar{m} = 80\hat{i} - 10\hat{k}$$

From Equation (4.68), we get

$${}^B f_x = \bar{n} \cdot \bar{f} = 0,$$

$${}^B f_y = \bar{o} \cdot \bar{f} = 0,$$

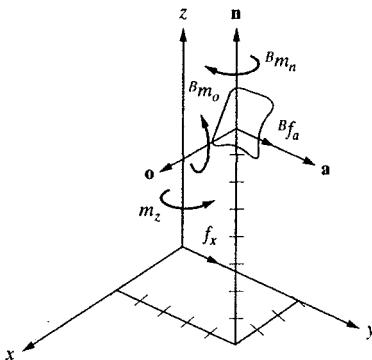


Figure 4.9 Equivalent force–moment systems in two different frames.

$${}^Bf_z = \bar{a} \cdot \bar{f} = 10 \rightarrow {}^Bf = [0, 0, 10],$$

$${}^Bm_x = \bar{n} \cdot [(\bar{f} \times \bar{p}) + \bar{m}] = -10,$$

$${}^Bm_y = \bar{o} \cdot [(\bar{f} \times \bar{p}) + \bar{m}] = 80,$$

$${}^Bm_z = \bar{a} \cdot [(\bar{f} \times \bar{p}) + \bar{m}] = 0 \rightarrow {}^Bm = [-10, 80, 0].$$

This means that a 10-lb force along the \bar{a} -axis of the frame B , together with two moments of -10 along the \bar{n} -axis and 80 along the \bar{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 4.9 shows the two equivalent force–moment systems.

4.7 DESIGN PROJECT

You may want to continue with the analysis of your robot. You will have to develop the dynamic equations of motion, which will enable you to calculate the power needed at each joint to move the robot at desired accelerations. This information will be used for choosing the appropriate actuators, as well as in controlling its motions.

Alternatively, 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 arms extended outwards. In this case, the second actuator acting on the second joint will experience the largest load. This estimate is not very accurate, since we are eliminating all coupling inertia terms, Coriolis accelerations, etc. But as was mentioned earlier, under low-load conditions, with low velocities, you can get a relatively acceptable estimate of the torques needed.

4.8 SUMMARY

In this chapter, we discussed the derivation of the dynamic equations of motion of the robots. These equations can be used to estimate the necessary 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 three-dimensional mechanisms with multiple degrees of freedom 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. For example, one 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, one 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 will discuss how a robot's motions are controlled and planned 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, New York, 1987.
3. Asada, Haruhiko, J. J. E., Slotine, *Robot Analysis and Control*, John Wiley and Sons, New York, 1986.
4. Sciacicco, Lorenzo, B. Siciliano, *Modeling and Control of Robot Manipulators*, McGraw-Hill, New York, 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, New York, 1988, pp. 329–347.
8. Pytel, Andrew, J. Kiusalaas, *Engineering Mechanics, Dynamics*, 2d Edition, Brooks/Cole Publishing, Pacific Grove, 1999.
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*, 2d Edition, Brooks/Cole Publishing, Pacific Grove, 1999.
12. Chicurel, Marina, "Once More. With Feeling," *Stanford Magazine*, March/April 2000, pp. 70–73.

PROBLEMS

1. Using Lagrangian mechanics, derive the equations of motion of a cart with two tires under the cart shown in Figure P.4.1.

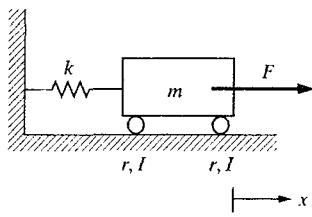


Figure P.4.1

2. Calculate the total kinetic energy of the link AB , attached to a roller with negligible mass, as shown in Figure P.4.2.

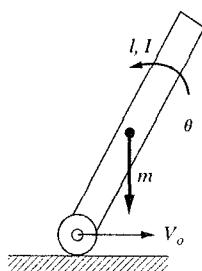


Figure P.4.2

3. Derive the equations of motion for the two-link mechanism with distributed mass shown in Figure P.4.3.

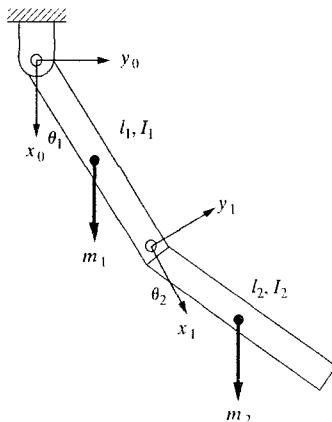


Figure P.4.3

4. Write the equations that express U_{62} , U_{35} , U_{53} , U_{h23} , and U_{534} for a six-axis cylindrical-RPY robot in terms of the A and Q matrices.
 5. Using Equations (4.44)–(4.49), write the equations of motion for a three-degree-of-freedom revolute robot, and explain what each term is.

6. Expand D_{134} and D_{15} of Equation (4.44) in terms of their constituent matrices.
 7. An object is subjected to the following forces and moments relative to the reference frame:

$$B = \begin{bmatrix} 0.707 & 0.707 & 0 & 2 \\ 0 & 0 & 1 & 5 \\ 0.707 & -0.707 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad F^T = [10, 0, 5, 12, 20, 0] \text{ N, N}\cdot\text{m}.$$

Attached to the object is a frame describing the orientation and the location of the object. Find the equivalent forces and torques acting on the object relative to the current frame.

8. To assemble two parts together, one part must be pushed into the other with a force of 10 lb in the x -axis direction, 5 lb in the y -direction, and be turned with a torque of 5 lb. in the x -axis direction. The object's location relative to the base frame of a robot is described by ${}^R T_O$:

$${}^R T_O = \begin{bmatrix} 0 & -0.707 & 0.707 & 4 \\ 1 & 0 & 0 & 6 \\ 0 & 0.707 & 0.707 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

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.

5

Trajectory Planning

5.1 INTRODUCTION

In the previous chapters, we studied the kinematics and dynamics of 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 in order to place the robot at a desired position and orientation, and with what 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 will study the sequence of movements that must be made to create a controlled movement between motion segments, in straight-line motions, or in sequential motions. Path and trajectory planning requires the use of both kinematics and dynamics of robots. In practice, precise motion requirements are so intensive that approximations are always necessary.

5.2 PATH VS. TRAJECTORY

A path is defined as a sequence of robot configurations in a particular order without regard to the timing of these configurations. So, if a robot goes from point (and thus, configuration) *A* to point *B* to point *C* in Figure 5.1, the sequence of the configurations between *A* and *B* and *C* constitutes a path [1]. However, a trajectory is concerned about when each part of the path must be attained, thus specifying timing. As a result, in Figure 5.1, regardless of when points *B* and *C* are reached, the path is the same, while as a trajectory, depending on the velocities and accelerations, points *B* and *C* may be reached at different times, creating different trajectories. In this chapter, we are not only concerned about the path a robot takes, but also its velocities and accelerations.

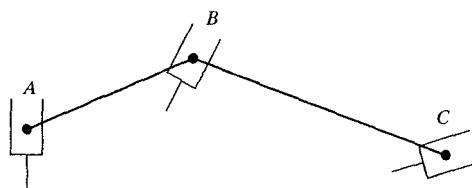


Figure 5.1 Sequential robot movements in a path.

5.3 JOINT-SPACE VS. CARTESIAN-SPACE DESCRIPTIONS

Consider a six-axis robot 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 Chapter 2, one 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 by its joint values is called *joint-space* description. In this case, although the robot will eventually move to the desired position, the motion between the two points is unpredictable, as will be seen later.

Now assume that a straight line is drawn between points *A* and *B* and that it is desirable to have the robot move from point *A* to point *B*, but it is also desirable that it follow the straight line between the two points. To do this, it will be necessary to divide the straight line into small portions, as shown in Figure 5.2, 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 the next segment.

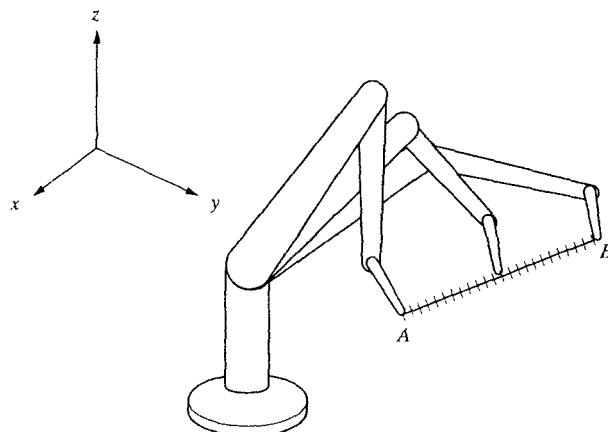


Figure 5.2 Sequential motions of a robot to follow a straight line.

When all segments are completed, the robot will be at point *B*, as desired. However, in this case, unlike the joint-space case previously mentioned, the motion is known at all times. The sequence of movements that the robot makes is described in *Cartesian space* and is converted to joint-space 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 advantages and disadvantages.

Cartesian-space trajectories are very easy to visualize. Since the trajectories are in the common Cartesian space in which we all operate, it is easy to visualize what the end-effector's trajectory must be. However, Cartesian space trajectories are computationally extensive, and require a faster processing time for similar resolution as 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 5.3(a). If not careful, one 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 [2]. This is true because it may be impossible to know whether the robot can actually make a particular location and orientation before the motion is made. Also, as shown in Figure 5.3(b), the motion between two points may require an instantaneous change in the joint values (in Chapter 2 we discussed why this may happen), which is impossible to make. Some of these problems may be solved by specifying via points through which the robot must pass in order to avoid obstacles and other similar singularities.

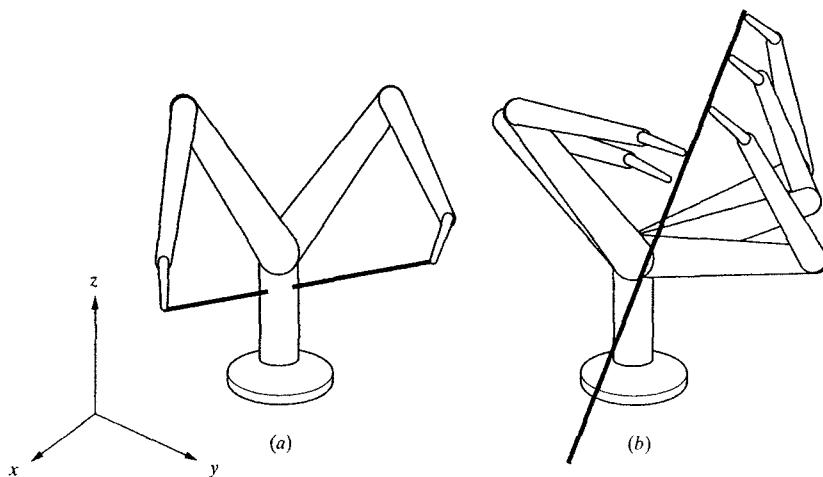


Figure 5.3 Cartesian-space trajectory problems: (a) The trajectory specified in Cartesian coordinates may force the robot to run into itself, and (b) the trajectory may require a sudden change in the joint angles.

5.4 BASICS OF TRAJECTORY PLANNING

To understand the basics of planning a trajectory in joint-space and Cartesian-space, let's consider a simple two-degree of freedom robot (mechanism). In this case, as shown in Figure 5.4, we desire 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 that 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 the maximum rate of 10 degrees/sec. One way to move the robot from point A to point 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 the figure. The trajectory of the end of the robot is also shown. As indicated, the path is irregular, and the distances traveled by the robot's end are not uniform.

Now suppose that the motions of both joints of the robot are normalized by a common factor such that the joint with smaller motion will move proportionally slower and that both joints will start and stop their motion simultaneously. In this case, both joints move at different speeds, but move continuously together. α will change 4 degrees per second while β changes 10 degrees per second. The resulting trajectory will be different, as shown in Figure 5.5. You notice that the segments of the movement are much more similar to each other than before, but that the path is still irregular (and different from the previous case). Both of these cases were planned in joint-space. The only calculation needed was the joint values for the destination and, in the second case, normalization of the joint velocities.

Now suppose we desire that the robot's hand 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 segments, and solve for necessary angles α and β at each point, as shown in Figure 5.6. This is called interpolation between points A and B [3,4,5]. You notice that in this case, the path is a straight line, but that the joint angles are not uniformly changing. Although the resulting motion is a straight (and thus, known) trajectory, it is necessary to solve for the joint

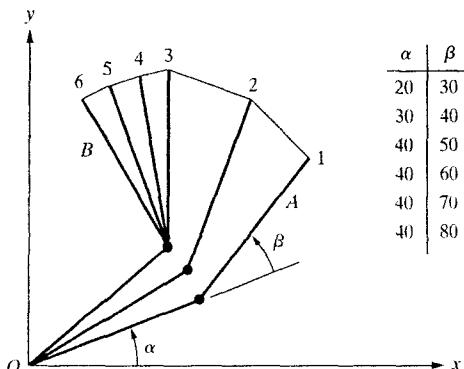


Figure 5.4 Joint-space, nonnormalized movements of a robot with two degrees of freedom.

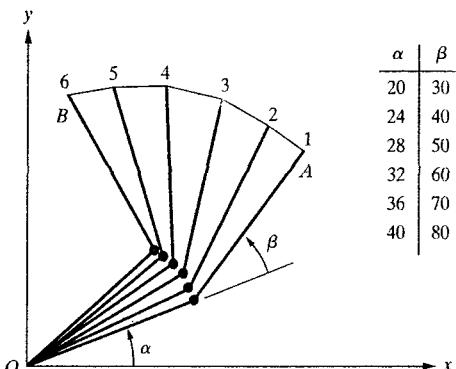


Figure 5.5 Joint-space, normalized movements of a robot with two degrees of freedom.

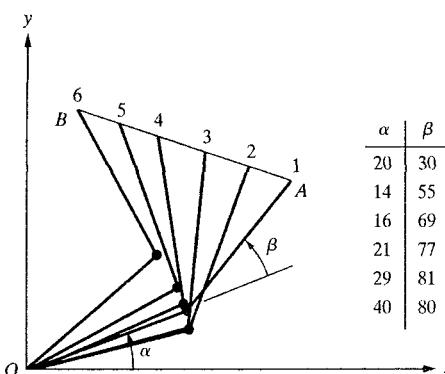


Figure 5.6 Cartesian-space movements of a two-degree-of-freedom robot.

values at each point. Obviously, many more points must be calculated for better accuracy, as with so few segments, the robot will not exactly follow the lines at each segment. This trajectory is in Cartesian-space, since all segments of the motion must be calculated based on the information expressed in a Cartesian frame.

In this case, it is assumed that the robot's actuators are strong enough to provide large forces necessary to accelerate and decelerate the joints as needed. For example, you notice that we are assuming that 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 be following a trajectory that is different from our assumption; it will be slightly behind as it accelerates to the desired speed. To improve the situation, we may actually divide the segments differently, such that at the beginning, we will move the arm at smaller segments as we speed up the arm; then we can go at a constant cruising rate and decelerate with smaller segments as we approach point B . This is schematically shown in Figure 5.7. 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, instead of dividing the straight line AB into equal segments, we will divide it based on $x = 1/2at^2$ until such time t

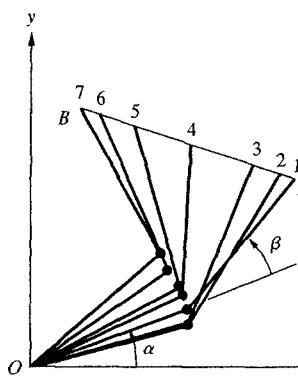


Figure 5.7 Trajectory planning with an acceleration-deceleration regimen.

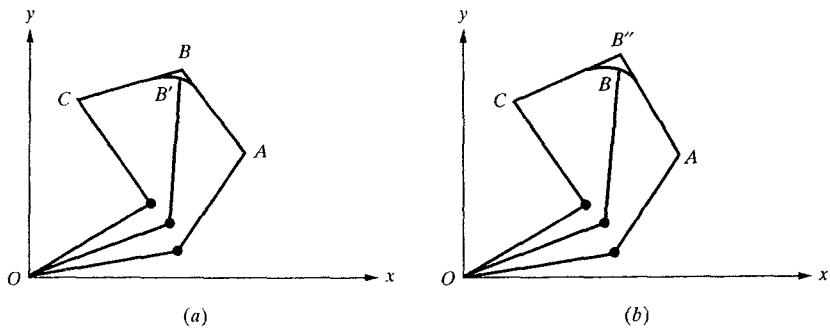


Figure 5.8 Blending of different motion segments in a path.

that 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 to this trajectory planning is to plan a path that is not straight, but that it follows some desired path (e.g., 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.

So far, we have *only* considered the movement of the robot between two points A and B . However, in most 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 movements.

Assume that the robot is to go from point A to point B and then to C , as shown in Figure 5.8. One way to run the robot is to accelerate from point A towards point B , cruise, and then decelerate and stop at point B , start again at point B and accelerate towards point C , cruise, and decelerate in order to stop at C . This stop-and-go motion will create jerky motions with unnecessary stops. An alternative way is to blend the two portions of the motion at point B , so that the robot will approach point B , decelerate if necessary, follow the blended path, accelerate once again towards point C , and eventually stop at C . This creates a much more graceful motion,

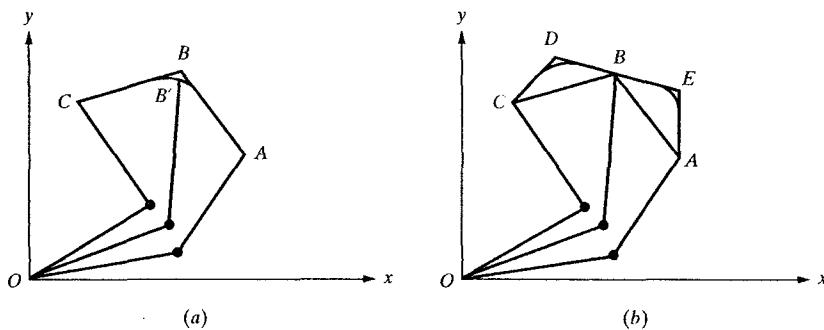


Figure 5.9 An alternative scheme for ensuring that the robot will go through a specified point during blending of motion segments. Two via points D and E are picked such that point B will fall on the straight-line section of the segment ensuring that the robot will pass through point B .

reduces the stresses on the robot, and requires less energy. If the motion consists of more segments, all intermediate segments can be blended together. Notice that due to this blending of the segments, the robot will go through a different point B' (Figure 5.8(a)) and not B . If it is necessary that the robot go through point B exactly, then a different point B'' must be specified such that after blending, the robot still goes through point B (Figure 5.8(b)). Another alternative [2] is to specify two via points D and E before and after point B , as shown in Figure 5.9, such that point B will fall on the straight-line portion of the motion between the via points and thus ensure that the robot will go through point B .

In the next sections, we will 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 later used to run the robot. This is commonly done for teaching robots in tasks such as spray painting of automobiles, in seam welding of complicated shapes, and in other similar tasks.

5.5 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. A number of different schemes, such as polynomials of different orders and linear functions with parabolic blends, can be used for this purpose. We next present a discussion of some of these schemes used in joint-space trajectory planning. Please notice that these schemes specify joint values and not Cartesian values.

5.5.1 Third-Order Polynomial Trajectory Planning

In this application, the initial location and orientation of the robot is known, and using the inverse kinematic equations, we find the final joint angles for the desired position and orientation. Now consider one of the joints, which at the beginning of the motion segment at time t_i is at θ_i . We desire to have the joint 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 that the velocities at the beginning and the end of the motion segment are zero (or other known values). These four pieces of information allow us to solve for four unknowns (or a third-order polynomial) in the form of

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3, \quad (5.1)$$

where the initial and final conditions are

$$\begin{aligned} \theta(t_i) &= \theta_i, \\ \theta(t_f) &= \theta_f, \\ \dot{\theta}(t_i) &= 0, \\ \dot{\theta}(t_f) &= 0. \end{aligned} \quad (5.2)$$

Taking the first derivative of the polynomial of Equation (5.1), we get

$$\dot{\theta}(t) = c_1 + 2c_2 t + 3c_3 t^2. \quad (5.3)$$

Substituting the initial and final conditions into Equations (5.1) and (5.3) yields

$$\begin{aligned} \theta(t_i) &= c_0 = \theta_i, \\ \theta(t_f) &= c_0 + c_1 t_f + c_2 t_f^2 + c_3 t_f^3, \\ \dot{\theta}(t_i) &= c_1 = 0, \\ \dot{\theta}(t_f) &= c_1 + 2c_2 t_f + 3c_3 t_f^2 = 0. \end{aligned} \quad (5.4)$$

By solving these four equations simultaneously, we get the necessary values for 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 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 nonzero, the given values can be used in these equations.

If more than two points are specified such that the robot will go through the points successively, the boundary 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 5.1

It is desired to have the first joint of a six-axis robot go from 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 Equation (5.4), 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 will result in the following cubic polynomial equation for position, as well as the velocity and acceleration equations:

$$\theta(t) = 30 + 5.4t^2 - 0.72t^3,$$

$$\dot{\theta}(t) = 10.8t - 2.16t^2,$$

$$\ddot{\theta}(t) = 10.8 - 4.32t.$$

Substituting the desired time intervals into the motion equation gives

$$\theta(1) = 34.68^\circ, \quad \theta(2) = 45.84^\circ, \quad \theta(3) = 59.16^\circ, \quad \theta(4) = 70.32^\circ.$$

The joint angles, velocities, and accelerations are shown in Figure 5.10. Notice that in this case, the acceleration needed at the beginning of the motion is 10.8 degrees/sec² (as well as -10.8 degrees/sec² deceleration at the conclusion of the motion).

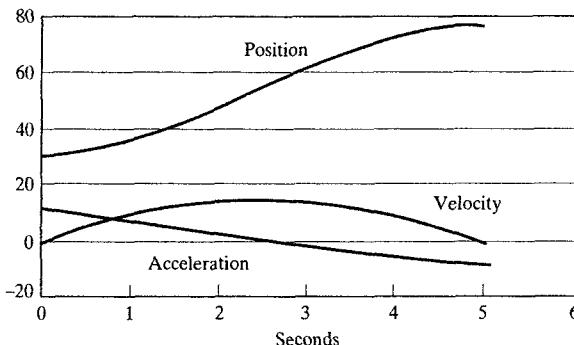


Figure 5.10 Joint positions, velocities, and accelerations for Example 5.1.

Example 5.2

Suppose that the robot arm of Example 5.1 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 what the position and velocity of the joint are. Using these values as initial conditions for the next segment, we get

$$\theta(t) = c_0 + c_1t + c_2t^2 + c_3t^3,$$

$$\dot{\theta}(t) = c_1 + 2c_2t + 3c_3t^2,$$

$$\ddot{\theta}(t) = 2c_2 + 6c_3t,$$

where

$$at_i = 0, \quad \theta_i = 75, \quad \dot{\theta}_i = 0,$$

$$at_f = 3, \quad \theta_f = 105, \quad \dot{\theta}_f = 0,$$

which yields

$$c_0 = 75, \quad c_1 = 0, \quad c_2 = 10, \quad c_3 = -2.222,$$

$$\theta(t) = 75 + 10t^2 - 2.222t^3,$$

$$\dot{\theta}(t) = 20t - 6.666t^2,$$

$$\ddot{\theta}(t) = 20 - 13.333t.$$

Figure 5.11 shows the position, velocity, and accelerations for the entire motion. As can be seen, the boundary conditions are as expected. However, you will 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. Whether or not the robot is capable of creating such accelerations 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 will be [4]

$$\left| \ddot{\theta} \right|_{\max} = \left| \frac{6(\theta_f - \theta_i)}{(t_f - t_i)^2} \right|,$$

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.

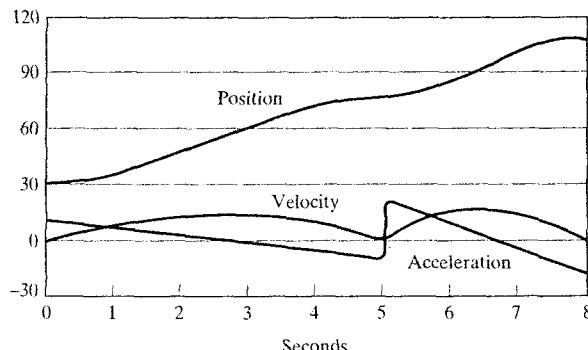


Figure 5.11 Joint positions, velocities, and accelerations for Example 5.2.

5.5.2 Fifth-Order Polynomial Trajectory Planning

In addition to specifying the initial and ending positions and velocities of a segment, it is also possible to specify initial and ending accelerations for a segment. In this case, the total number of boundary conditions is six, enabling us to use a fifth-order polynomial to plan a trajectory:

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5, \quad (5.5)$$

$$\dot{\theta}(t) = c_1 + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4, \quad (5.6)$$

$$\ddot{\theta}(t) = 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3. \quad (5.7)$$

These equations allow us to calculate the coefficients of a fifth-order polynomial with position, velocity, and acceleration boundary conditions.

Example 5.3

Repeat Example 5.1, but this time assume that the initial acceleration and final deceleration will be 5 degrees/sec².

Solution From Example 5.1 and the given accelerations, we have

$$\theta_i = 30^\circ, \quad \dot{\theta}_i = 0 \text{ degrees/sec}, \quad \ddot{\theta}_i = 5 \text{ degrees/sec}^2,$$

$$\theta_f = 75^\circ, \quad \dot{\theta}_f = 0 \text{ degrees/sec}, \quad \ddot{\theta}_f = -5 \text{ degrees/sec}^2$$

Using Equations (5.5) through (5.7), with the given initial and final boundary conditions, we get

$$c_0 = 30, \quad c_1 = 0, \quad c_2 = 2.5,$$

$$c_3 = 1.6, \quad c_4 = -0.58, \quad c_5 = 0.0464.$$

This results in the following motion equations:

$$\theta(t) = 30 + 2.5t^2 + 1.6t^3 - 0.58t^4 + 0.0464t^5,$$

$$\dot{\theta}(t) = 5t + 4.8t^2 - 2.32t^3 + 0.232t^4,$$

$$\ddot{\theta}(t) = 5 + 9.6t - 6.96t^2 + 0.928t^3.$$

Figure 5.12 shows the position, velocity, and acceleration graphs for the joint. The maximum acceleration is 8.7 degrees/sec².

5.5.3 Linear Segments with Parabolic Blends

Another alternative for joint-space trajectory planning is to run the joints at constant speed between the initial and final locations, as was discussed in Section 5.4. This is equivalent of a first order polynomial, where the velocity is constant and acceleration is zero. However, this also means that at the beginning and 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 the end of the motion segment, creating

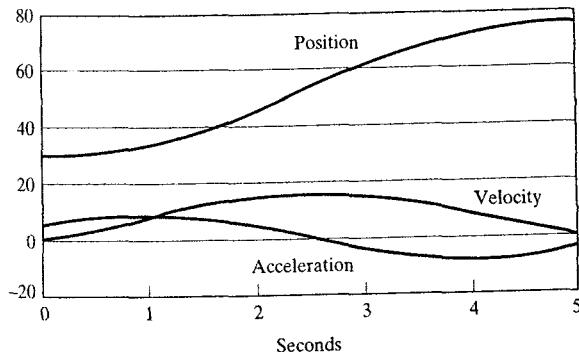


Figure 5.12 Joint position, velocity, and acceleration for Example 5.3.

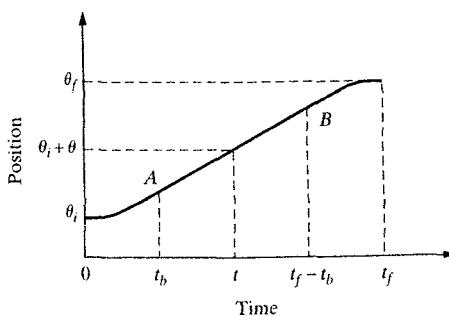


Figure 5.13 Scheme for linear segments with parabolic blends.

continuous position and velocity, as shown in Figure 5.13. Assuming that the initial and 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

$$\begin{aligned}\theta(t) &= c_0 + c_1 t + \frac{1}{2} c_2 t^2, \\ \dot{\theta}(t) &= c_1 + c_2 t, \\ \ddot{\theta}(t) &= c_2.\end{aligned}\tag{5.8}$$

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 gives parabolic segments in the form

$$\theta(t) = \theta_i + \frac{1}{2} c_2 t^2, \quad (5.9)$$

$$\dot{\theta}(t) = c_2 t, \quad (5.10)$$

$$\ddot{\theta}(t) = c_2. \quad (5.11)$$

Clearly, for the linear segment, the velocity will be 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 Equation (5.10), we find the joint positions and velocities for points A , B and the final point as follows:

$$\begin{aligned} \theta_A &= \theta_i + \frac{1}{2} c_2 t_b^2, \\ \dot{\theta}_A &= c_2 t_b = \omega, \\ \theta_B &= \theta_A + \omega((t_f - t_b) - t_b) = \theta_A + \omega(t_f - 2t_b), \\ \dot{\theta}_B &= \dot{\theta}_A = \omega, \\ \theta_f &= \theta_B + (\theta_A - \theta_i), \\ \dot{\theta}_f &= 0. \end{aligned} \quad (5.12)$$

The necessary blending time t_b can be found from Equations (5.12) as follows:

$$\begin{cases} c_2 = \frac{\omega}{t_b}, \\ \theta_f = \theta_i + c_2 t_b^2 + \omega(t_f - 2t_b), \end{cases} \rightarrow \begin{cases} \theta_f = \theta_i + \left(\frac{\omega}{t_b}\right) t_b^2 + \omega(t_f - 2t_b). \end{cases} \quad (5.13)$$

From Equation (5.13), we calculate the blending time as

$$t_b = \frac{\theta_i - \theta_f + \omega t_f}{\omega}. \quad (5.14)$$

Obviously, the time t_b cannot be bigger than half of the total time t_f , which results in a parabolic speedup and a parabolic slowdown, with no linear segment. A corresponding maximum velocity of $\omega_{max} = 2(\theta_f - \theta_i)/t_f$ can be found from Equation (5.14). 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 thus can be expressed as follows:

$$\theta(t) = \theta_f - \frac{1}{2} c_2 (t_f - t)^2, \quad \text{where } c_2 = \frac{\omega}{t_b}$$

$$\rightarrow \begin{cases} \theta(t) = \theta_f - \frac{\omega}{2t_b}(t_f - t)^2, \\ \dot{\theta}(t) = \frac{\omega}{t_b}(t_f - t), \\ \ddot{\theta}(t) = -\frac{\omega}{t_b}. \end{cases} \quad (5.15)$$

Example 5.4

Joint 1 of the six-axis robot of Example 5.1 is to go from initial angle of $\theta_i = 30^\circ$ to the final angle of $\theta_f = 70^\circ$ in 5 seconds with a cruising velocity of $\omega_1 = 10$ degrees/second. Find the necessary time for blending, and plot the joint positions, velocities, and accelerations.

Solution From Equations (5.9)–(5.11), (5.14), and (5.15), we get the following (see Figure 5.14):

$$t_c = \frac{\theta_i - \theta_f + \omega_1 t_f}{\omega_1} = \frac{30 - 70 + 10(5)}{10} = 1 \text{ sec.}$$

For $\theta = \theta_i$ to θ_A	For $\theta = \theta_A$ to θ_B	For $\theta = \theta_B$ to θ_f
$\begin{cases} \theta = 30 + 5t^2, \\ \dot{\theta} = 10t, \\ \ddot{\theta} = 10. \end{cases}$	$\begin{cases} \theta = \theta_A + 10t, \\ \dot{\theta} = 10, \\ \ddot{\theta} = 0. \end{cases}$	$\begin{cases} \theta = 70 - 5(5 - t)^2, \\ \dot{\theta} = 10(5 - t), \\ \ddot{\theta} = -10. \end{cases}$

Figure 5.14 shows the position, velocity, and acceleration graphs for this joint.

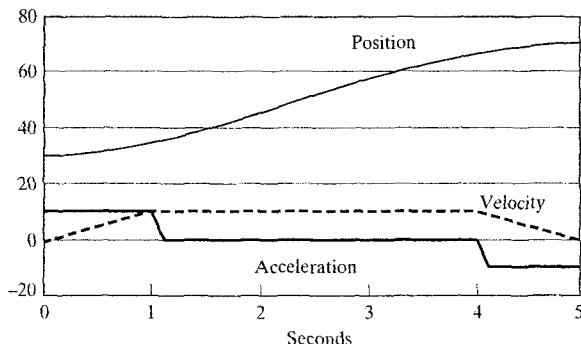


Figure 5.14 (a) Position, (b) velocity, and (c) acceleration graphs for Joint 1 of Example 5.4.

5.5.4 Linear Segments with Parabolic Blends and Via Points

Suppose that 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 was 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 will use the boundary conditions of each point to calculate the coefficients of the parabolic segments. For 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 thus, 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.

5.5.5 Higher Order Trajectories

When, in addition to the initial and final destination points, other via points (including a liftoff and a setdown point) are specified, we may match the position, velocity, and accelerations of the two segments at each point to plan a continuous trajectory. Incorporating the initial and final boundary conditions together with this information enables us to use higher order polynomials in the form

$$\theta(t) = c_0 + c_1t + c_2t^2 + \dots + c_{n-1}t^{n-1} + c_nt^n, \quad (5.16)$$

so that the trajectory will pass 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., liftoff), a third-order polynomial is used to plan a trajectory between two via points (e.g., liftoff and setdown), and another fourth-order polynomial is used to plan the trajectory between the last via point (e.g., setdown) 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.

To see how these may work, let's take a closer look at a 4–3–4 trajectory. A fourth-order polynomial will have five unknown coefficients in it. Similarly, a third-order polynomial will have four unknown coefficients. Thus, a 4–3–4 trajectory will have a total of 14 unknown coefficients in the following form:

$$\begin{aligned}\theta(t)_1 &= a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4, \\ \theta(t)_2 &= b_0 + b_1t + b_2t^2 + b_3t^3, \\ \theta(t)_3 &= c_0 + c_1t + c_2t^2 + c_3t^3 + c_4t^4.\end{aligned}\quad (5.17)$$

However, there are also 14 boundary and blending conditions available that can be used to solve for all unknown coefficients and plan the trajectory:

- (1) Initial position of θ_1 is known.
- (2) Initial velocity can be specified.
- (3) Initial acceleration can 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 must be the same as the initial position of 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 subsequent) via point θ_n is specified and is the same as the final position of the third-order segment.
- (9) The position of the second (and subsequent) via point must be 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.

We will denote the global, normalized, time variable for the whole motion as t , while τ_j denotes specific local times for each segment j . We will also assume that the local initial starting time τ_{ji} for each segment is zero and that 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 the preceding assumptions and data, 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 . Thus,

$$\theta_1 = a_0. \quad (5.18)$$

- (2) The starting velocity at $\tau_1 = 0$ for the first segment is known. Thus,

$$\dot{\theta}_1 = a_1. \quad (5.19)$$

- (3) The starting acceleration at $\tau_1 = 0$ for the first segment is known. Thus,

$$\ddot{\theta}_1 = 2a_2. \quad (5.20)$$

- (4) Position of the first via point θ_2 at the conclusion of the first segment at local time τ_{1f} is known. Thus,

$$\theta_2 = a_0 + a_1(\tau_{1f}) + a_2(\tau_{1f})^2 + a_3(\tau_{1f})^3 + a_4(\tau_{1f})^4. \quad (5.21)$$

- (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,

$$\theta_2 = b_0. \quad (5.22)$$

- (6) Continuous velocity must be maintained at the via point. Thus,

$$a_1 + 2a_2(\tau_{1f}) + 3a_3(\tau_{1f})^2 + 4a_4(\tau_{1f})^3 = b_1. \quad (5.23)$$

- (7) Continuous acceleration must be maintained at the via point. Thus,

$$2a_2 + 6a_3(\tau_{1f}) + 12a_4(\tau_{1f})^2 = 2b_2. \quad (5.24)$$

- (8) The position of a second via point θ_3 at the conclusion of the third-order segment at time τ_{2f} is specified. Thus,

$$\theta_3 = b_0 + b_1(\tau_{2f}) + b_2(\tau_{2f})^2 + b_3(\tau_{2f})^3. \quad (5.25)$$

- (9) The position of the via point θ_3 must be the same as the initial position of the next segment at $\tau_3 = 0$ for continuity. Thus,

$$\theta_3 = c_0. \quad (5.26)$$

- (10) Continuous velocity must be maintained at the via point. Thus,

$$b_1 + 2b_2(\tau_{2f}) + 3b_3(\tau_{2f})^2 = c_1. \quad (5.27)$$

- (11) Continuous acceleration must be maintained at the via point. Thus,

$$2b_2 + 6b_3(\tau_{2f}) = 2c_2. \quad (5.28)$$

- (12) The position of destination at the conclusion of the last segment τ_{3f} is specified as θ_f . Thus,

$$\theta_4 = c_0 + c_1(\tau_{3f}) + c_2(\tau_{3f})^2 + c_3(\tau_{3f})^3 + c_4(\tau_{3f})^4. \quad (5.29)$$

- (13) Velocity of the destination at the conclusion of the last segment at time τ_{3f} is specified. Thus,

$$\dot{\theta}_4 = c_1 + 2c_2(\tau_{3f}) + 3c_3(\tau_{3f})^2 + 4c_4(\tau_{3f})^3. \quad (5.30)$$

(14) Acceleration of the destination at the conclusion of the last segment at time τ_{3f} is specified. Thus,

$$\ddot{\theta}_4 = 2c_2 + 6c_3(\tau_{3f}) + 12c_4(\tau_{3f})^2. \quad (5.31)$$

Equations (5.18) through (5.31) can be rewritten in a matrix form as

$$\begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \\ 0 \\ 0 \\ 0 \\ 0 \\ \theta_3 \\ \dot{\theta}_3 \\ \ddot{\theta}_3 \\ 0 \\ 0 \\ 0 \\ \theta_4 \\ \dot{\theta}_4 \\ \ddot{\theta}_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \tau_{1f} & \tau_{1f}^2 & \tau_{1f}^3 & \tau_{1f}^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2\tau_{1f} & 3\tau_{1f}^2 & 4\tau_{1f}^3 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 6\tau_{1f} & 12\tau_{1f}^2 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \theta_3 & 0 & 0 & 0 & 0 & 0 & 1 & \tau_{2f} & \tau_{2f}^2 & \tau_{2f}^3 & 0 & 0 & 0 & 0 & 0 \\ \dot{\theta}_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2\tau_{2f} & 3\tau_{2f}^2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6\tau_{2f} & 0 & 0 & -2 & 0 & 0 \\ \theta_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \tau_{3f} & \tau_{3f}^2 & \tau_{3f}^3 & \tau_{3f}^4 \\ \dot{\theta}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2\tau_{3f} & 3\tau_{3f}^2 & 4\tau_{3f}^3 \\ \ddot{\theta}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6\tau_{3f} & 12\tau_{3f}^2 & 0 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}, \quad (5.32)$$

or

$$[\theta] = [M][C]$$

and

$$[C] = [M]^{-1}[\theta] \quad (5.33)$$

The unknown coefficients can be found from Equation (5.33) by calculating $[M]^{-1}$. Thus, the equations of motion for the three segments are known and the robot can be driven through the specified positions. 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 5.5

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. Determine the trajectory equations and plot the position, velocity, and acceleration curves for the joint given the following positions, velocities, and time duration for the three segments for one of the joints:

$$\begin{aligned} \theta_1 &= 30^\circ, & \dot{\theta}_1 &= 0, & \ddot{\theta}_1 &= 0, & \tau_{1i} &= 0, & \tau_{1f} &= 2, \\ \theta_2 &= 50^\circ, & \tau_{2i} &= 0, & \tau_{2f} &= 4, \\ \theta_3 &= 90^\circ, & \tau_{3i} &= 0, & \tau_{3f} &= 2, \\ \theta_4 &= 70^\circ, & \dot{\theta}_4 &= 0, & \ddot{\theta}_4 &= 0. \end{aligned}$$

Solution We can calculate the unknown coefficients of the three segments by substituting the given values directly into the matrices of Equation (5.32), or into Equations (5.18) through (5.31) and solving the resulting set of equations. The result is

$$\begin{array}{lll} a_0 = 30, & b_0 = 50, & c_0 = 90, \\ a_1 = 0, & b_1 = 20.477, & c_1 = -13.81, \\ a_2 = 0, & b_2 = 0.714, & c_2 = -9.286, \\ a_3 = 4.881, & b_3 = -0.833, & c_3 = 9.643, \\ a_4 = -1.191, & & c_4 = -2.024. \end{array}$$

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 5.15 shows the position, velocity, and acceleration of this joint for the given motion based on the 4–3–4 trajectory.

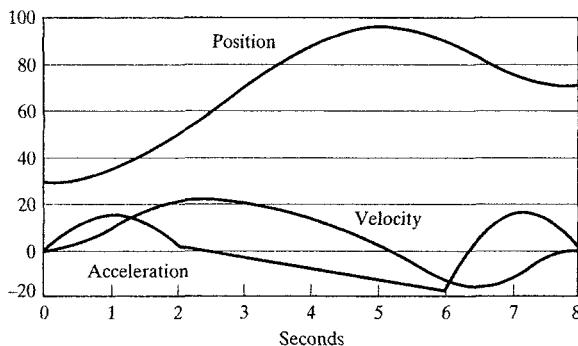


Figure 5.15 The position, velocity, and acceleration curves for the motion of the joint in Example 5.5, based on a 4–3–4 trajectory.

5.5.6 Other Trajectories

Many other schemes may also be used to plan trajectories. This includes bang-bang trajectories, square and trapezoidal acceleration profile trajectories, and sine function trajectories. Additionally, it is possible to use 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 the chapter.

5.6 CARTESIAN-SPACE TRAJECTORIES

As discussed through simple examples in Section 5.4, Cartesian-space trajectories relate to the motions of a robot relative to the Cartesian reference frame, as fol-

lowed 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 can be used for Cartesian-space trajectories, too. 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 of the motions for multiple destinations (such as via points) are 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:

$$\begin{aligned} T_f &= T_i R, \\ T_i^{-1} T_f &= T_i^{-1} T_i R, \\ R &= T_i^{-1} T_f. \end{aligned} \tag{5.34}$$

At least three different alternatives may be used to convert this transformation into small segments:

- (1) Since we desire to have a smooth straight-line transformation between the initial and final locations, we should like to have large number of very small segments. This, in reality, creates a large number of differential motions [3]. Using the equations that were 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

$$D = JD_\theta \quad \text{and} \quad D_\theta = J^{-1}D,$$

$$dT = \Delta \cdot T,$$

$$T_{\text{new}} = T_{\text{old}} + dT.$$

This technique requires extensive calculations and only works if the inverse Jacobian exists.

- (II) 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.
- (III) The transformation between the initial and final locations R can be decomposed into a translation and one rotation about an axis \hat{k} . 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 [2,3,6]. Both transformations are executed simultaneously (Figure 5.16).

For more information about Cartesian-space trajectory planning, please refer to the references at the end of this chapter.

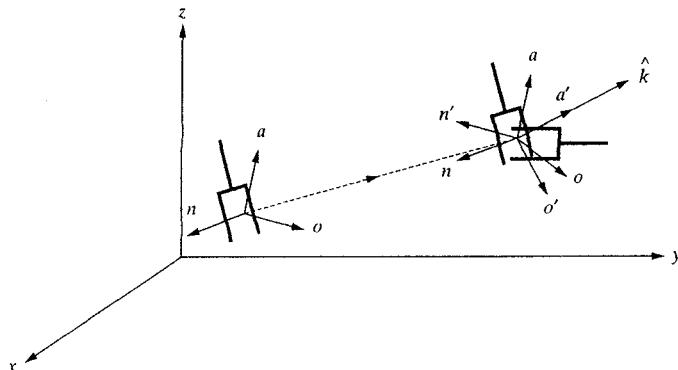


Figure 5.16 Transformation between an initial and final locations in Cartesian-space trajectory planning. The motion can be decomposed into a translation and a rotation about an axis \hat{k} .

Example 5.6

A two-degree-of-freedom planar robot is to follow a straight line between the start (3,10) and the end (8,14) points 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

$$\frac{y - 14}{x - 8} = \frac{14 - 10}{8 - 3} = 0.8,$$

or

$$y = 0.8x + 7.6.$$

The coordinates of the intermediate points can be found by simply dividing the differences between the x 's and the y 's of the start and the end points. The angles for the two joints are then found for each intermediate point. The results are shown in Table 5.1 and Figure 5.17.

TABLE 5.1 THE COORDINATES AND THE JOINT ANGLES FOR EXAMPLE 5.6.

#	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

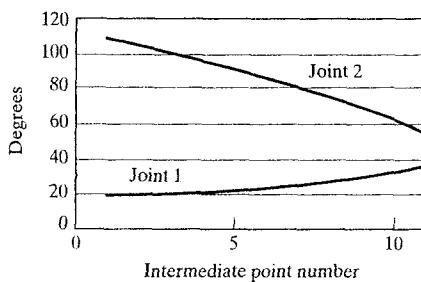


Figure 5.17 The joint positions for the robot of Example 5.6.

Example 5.7

A three-degree-of-freedom robot designed for lab experimentation at Cal Poly has two links, each 9 inches long. As shown in Figure 5.18, the coordinate frames of the joints are such that when all angles are zero, the arm is pointed upwards. We desire 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 if the inverse kinematic equations of the robot are as follows:

$$\theta_1 = \tan^{-1}(-P_x/P_y),$$

$$\theta_3 = \cos^{-1}\left[\left((P_y/C_1)^2 + (P_z)^2 - 162\right)/162\right],$$

$$\theta_2 = \cos^{-1}\left[(P_z C_1(1 + C_3) + P_y S_3)/(18(1 + C_3)C_1)\right].$$

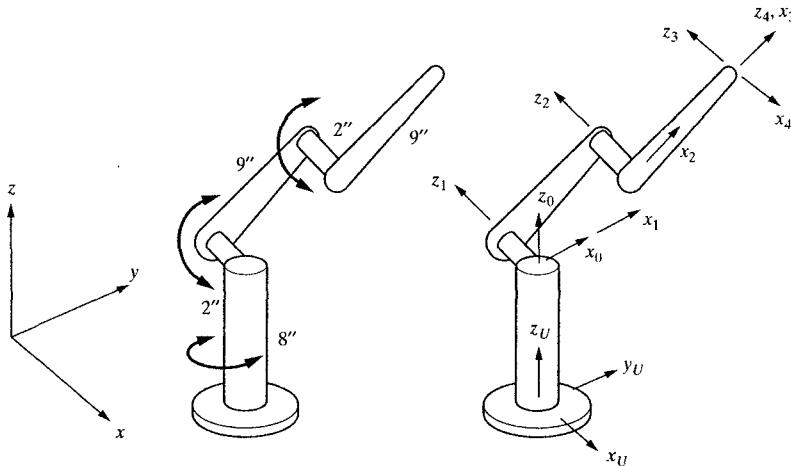


Figure 5.18 Robot of Example 5.7 and its coordinate frames.

Solution In this solution, we will divide the distance between the start and the 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 the end points into 10 equal parts. The inverse kinematic equation is then used to calculate the joint angles for each intermediate point, as shown in Table 5.2. The joint angles are shown in Figure 5.19.

TABLE 5.2 THE HAND-FRAME COORDINATES AND JOINT ANGLES FOR THE ROBOT OF EXAMPLE 5.7

X	Y	Z	θ_1	θ_3	θ_2
9	6	10	56.3	104.7	27.2
8.4	5.9	9.8	54.9	109.2	25.4
7.8	5.8	9.6	53.4	113.6	23.8
7.2	5.7	9.4	51.6	117.9	22.4
6.6	5.6	9.2	49.7	121.9	21.2
6	5.5	9	47.5	125.8	20.1
5.4	5.4	8.8	45	129.5	19.3
4.8	5.3	8.6	42.2	133	18.7
4.2	5.2	8.4	38.9	136.3	18.4
3.6	5.1	8.2	35.2	139.4	18.5
3	5	8	31	142.2	18.9

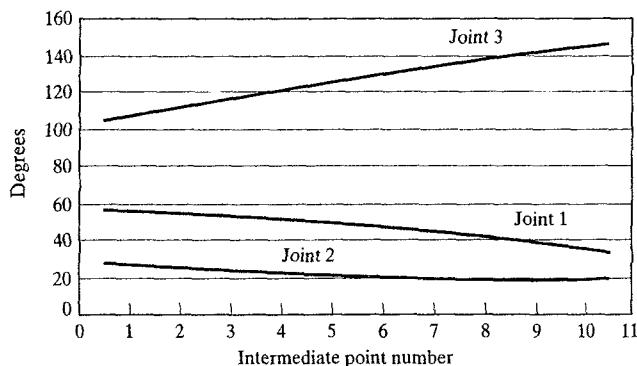


Figure 5.19 Joint angles for Example 5.7.

5.7 CONTINUOUS TRAJECTORY RECORDING

In some operations (such as spray painting and deburring), the motions required to accomplish a task may either be 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 that is required to accomplish a task in real time. This can be done by either releasing the joint brakes and physically moving the robot or by moving the joints of a robot model that is similar to the real robot, but is much lighter and can easily be moved. 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 will be forced to follow the same trajectory that was recorded, and thus, the robot will perform the task as planned.

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 reprogrammed. This is particularly difficult for large, massive, and heavy robots, especially if they are larger than a human operator.

5.8 DESIGN PROJECT

You may continue with the 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 your robot and run it. In the next chapter, we will discuss actuators, which will 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. For 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 small number of sections and continue with increasingly more sections 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 just a mechanism that moves in space.

5.9 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, and it may collide with other objects, may go through undesirable points, or may be inaccurate.

Trajectories may be planned in joint-space or in Cartesian-space. 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 easier, 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 realistic motions.

In the next chapter, we will discuss how a robot may be powered and actuated.

REFERENCES

1. Brady, M., J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason, editors, *Robot Motion: Planning and Control*, MIT Press, Cambridge, Mass., 1982.
2. Craig, John J., *Introduction to Robotics: Mechanics and Control*, 2d Edition, Addison Wesley, 1989.
3. Eman, K.F., Soo-Hun Lee, and J. C. Cesareone, "Trajectories," *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," *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, Englewood Cliffs, N.J., 1992.
6. Fu, K. S., R. C. Gonzales, and 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. Shahinpoor, Mohsen. *A Robot Engineering Textbook*, Harper and Row, Publishers, New York, 1987.
9. Snyder, Wesley, *Industrial Robots: Computer Interfacing and Control*, Prentice-Hall, Englewood Cliffs, N.J., 1985.

10. Kudo, Makoto, Y. Nasu, K. Mitobe, and B. Borovac, "Multi-arm Robot Control System for Manipulation of Flexible Materials in Sewing Operations," *Mechatronics*, Vol. 10, No. 3, pp. 371–402.
11. "Path-Planning Program for a Redundant Robotic Manipulator," *NASA Tech Briefs*, July 2000, pp. 61–62.
12. 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

1. It is desired to have the first joint of a 6-axis robot go from initial angle of 50° to a final angle of 80° in 3 seconds. Calculate the coefficients for a third order polynomial joint-space trajectory. Determine the joint angles, velocities, and accelerations at 1, 2, and 3 seconds. It is assumed that the robot starts from rest and stops at its destination.
2. It is desired to have the third joint of a six-axis robot go from 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 is supposed to have a final velocity of 5 degrees/sec.
3. The second joint of a six-axis robot is to go from initial angle of 20° to an intermediate angle of 80° in 5 seconds and then continue to its destination to 25° in another 5 seconds. Calculate the coefficients for third-order polynomials in joint-space. Plot the joint angles, velocities, and accelerations. Assume joint stops at intermediate point.
4. 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 allow a joint to go from 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 degrees/sec².
5. Joint 1 of a six-axis robot is to go from initial angle of $\theta_i = 30^\circ$ to the final angle of $\theta_f = 120^\circ$ in 4 seconds with a cruising velocity of $\omega_1 = 30$ degrees/sec. Find the necessary blending time for a trajectory with linear segment and parabolic blends, and plot the joint positions, velocities, and accelerations.
6. 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 below. Determine the trajectory equations and plot the position, velocity, and acceleration curves for the joint.

$$\begin{aligned}\theta_1 &= 20^\circ, & \dot{\theta}_1 &= 0, & \ddot{\theta}_1 &= 0, & \tau_{1i} &= 0, & \tau_{1f} &= 1, \\ \theta_2 &= 60^\circ, & \tau_{2i} &= 0, & \tau_{2f} &= 2, \\ \theta_3 &= 100^\circ, & \tau_{3i} &= 0, & \tau_{3f} &= 1, \\ \theta_4 &= 40^\circ, & \dot{\theta}_4 &\approx 0, & \ddot{\theta}_4 &= 0.\end{aligned}$$

7. A two-degree-of-freedom planar robot is to follow a straight line in Cartesian-space between the start (2,6) and the end (12,3) points 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.
8. The three-degree-of-freedom robot of Example 5.7, as shown in Figure 5.18, 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.

6

Actuators

6.1 INTRODUCTION

Actuators are the muscles of robots. If you imagine that the links and the joints are the skeleton of the robot, the actuators act as muscles, which move or rotate the links to change the configuration of robots. The actuator must have enough power to accelerate and decelerate the links and to carry the loads, 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. The following types are noteworthy:

- Electric motors
 - Servomotors
 - Stepper motors
 - Direct-drive electric motors
- Hydraulic actuators
- Pneumatic actuators
- Shape memory metal actuators
- Magnetostrictive actuators

Electric motors — especially 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 not used in new robots as often any more. Pneumatic cylinders are used in robots that have 1/2 degree of freedom, on-off type joints, as well as for insertion purposes. Direct drive electric motors, the shape memory metal type-actuators, and others like them are mostly in research and development stage and may become more useful in the near future.

In the next section, we will compare the common characteristics of different types of actuators, and then we will study each type individually.

6.2 CHARACTERISTICS OF ACTUATING SYSTEMS

The characteristics presented next may be used to compare different actuating systems. In addition to these, depending on the special circumstances in which they will be used, other characteristics may play a role in the design of robots. Examples include underwater systems, where waterproof operation of a system is very important, or space systems, where the liftoff weight and reliability are of absolute importance.

6.2.1 Weight, Power-to-Weight Ratio, Operating Pressure

It is important to consider the weight of the actuating system, as well as its 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 thus have a lower power-to-weight ratio. The higher the voltage of an electric motor, the better power-to-weight ratio it has. Pneumatic cylinders deliver the lowest 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. One is the hydraulic actuator, and the other is the hydraulic power unit. The system's power unit consists of a pump, which generates the high pressure needed to operate the cylinders and rams, a reservoir, filters, electric drive motors to drive the pump, cooling units, valves, etc. The actuators' role is only to move the joints. However, the power unit is normally stationary and located somewhere away from the robot itself. The power is brought to the robot via an umbilical tether hose. Thus, the actual power-to-weight ratio of the cylinders is very high for the moving parts. However, the power unit, which is very heavy, does not move and is not counted in this ratio. If the power unit must also move with the robot, the total power-to-weight ratio will be much less.

The power that the hydraulic system delivers is also very high, due to high operating pressures. This may range from 55 psi to 5,000 psi pressures. Pneumatic cylinders normally operate around 100 to 120 psi. The higher pressures in hydraulic systems mean higher powers, but they also require higher maintenance, and if a leak occurs, they can become more dangerous.

6.2.2 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 the resistance of wine against compression in a bottle during corking operation. The stiffer the system, the larger the load that is needed to deform it. Conversely, the more compliant the system, the easier it deforms under the load.

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, which is very high. As a result, hydraulic systems are very stiff and noncompliant. Conversely, pneumatic systems are easily compressed, and, thus, 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, thus, will be inaccurate. Similarly, if a small driving force is applied to a hydraulic ram, due to its stiffness, it will respond more rapidly and more accurately than a pneumatic system, which can deform under the same load. Additionally, the stiffer the system, the less it gives or deforms under load, and thus the more accurately it holds its position. Now consider a robot that is used to insert an integrated circuit chip into a circuit board. If the system is not stiff enough, the robot will not be able to push the chip into the board, since 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 cannot give enough to prevent damage to the robot or the part, whereas a compliant system will give 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. Thus, a working balance is needed between these two competing characteristics.

6.2.3 Use of Reduction Gears

Some systems, such as hydraulic devices, produce very large forces with short strokes. This means that the hydraulic ram may be moved very slightly while delivering its full force. As a result, there is no need to use reduction gear trains 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, which simplifies the design, reduces the weight and cost and rotating inertia of joints, reduces backlash, increases the reliability of the system, due to simpler design and fewer parts, and also reduces 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 to decrease their speed, as no one would want a robot arm to be rotating at such speeds. This, of course, increases the cost, number of parts, backlash, inertia of the rotating body, etc., as was mentioned earlier, but also increases the resolution of the system, as it is possible to rotate the link a very small angle.

Now suppose that, through a set of reduction gears with a reduction ratio of N , a load with inertia I_l , is connected to a motor with inertia I_m (including the inertia of the reduction gears), as shown in Figure 6.1. The torque and speed ratio between the motor and the load will be:

$$T_l = NT_m, \quad (6.1)$$

$$\dot{\theta}_l = \frac{1}{N} \dot{\theta}_m \quad \text{and} \quad \ddot{\theta}_l = \frac{1}{N} \ddot{\theta}_m.$$

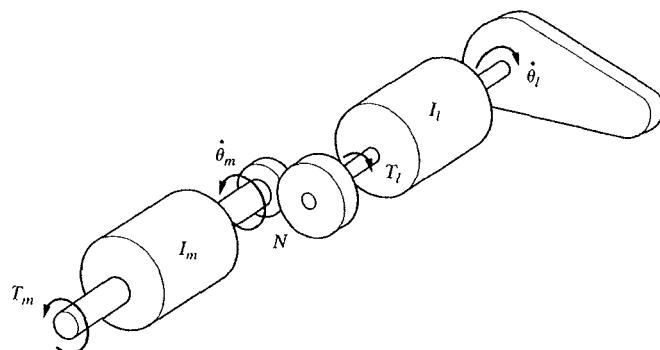


Figure 6.1 Inertia and torque relationship between a motor and a load.

If we write the torque balance equation for the system, we get

$$\begin{aligned} T_m &= I_m \ddot{\theta}_m + b_m \dot{\theta}_m + \frac{1}{N} T_l = I_m \ddot{\theta}_m + b_m \dot{\theta}_m + \frac{1}{N} (I_l \ddot{\theta}_l + b_l \dot{\theta}_l) \\ &= I_m \ddot{\theta}_m + b_m \dot{\theta}_m + \frac{1}{N^2} (I_l \ddot{\theta}_m + b_l \dot{\theta}_m), \end{aligned} \quad (6.2)$$

where b_m and b_l are viscous coefficients of friction for the motor and the load. However, as Equation 6.2 indicates, the effective inertia of the load felt by the motor is inversely proportional to the square of the reduction gear ratio, or

$$I_{\text{Effective}} = \frac{1}{N^2} I_l \quad \text{and} \quad I_{\text{Total}} = \frac{1}{N^2} I_l + I_m. \quad (6.3)$$

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 with it). For example, suppose that a reduction ratio of 10 is used in conjunction with a joint. The total inertia that the motor will see is only 1/100th the actual inertia, and thus the motor can accelerate quickly. In direct-drive systems, both electric and hydraulic, the motors 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.

Example 4.1

A motor with rotor inertia of 0.015 Kg m^2 and maximum torque of 8 N m is connected to a uniformly distributed arm with a concentrated mass at its end, as shown in Figure 6.2. 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 and (b) 30.

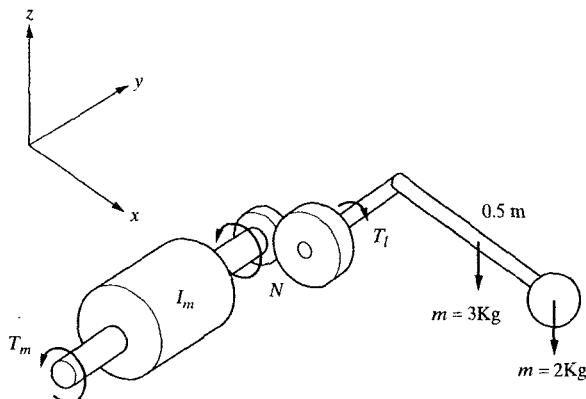


Figure 6.2 Schematic drawing of the system of Example 6.1.

Solution This is very similar to a robot arm and 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 Equation (6.3), we get the following:

$$(a) \quad I_{\text{Total}} = \frac{1}{N^2} I_l + I_m = \frac{1}{9} (0.75) + 0.015 = 0.098 \text{ Kgm}^2,$$

$$(b) \quad I_{\text{Total}} = \frac{1}{900} (0.75) + 0.015 = 0.0158 \text{ Kgm}^2.$$

As you can see, the total inertia with the higher gear reduction ratio is practically the same as the rotor inertia of the motor. The maximum angular accelerations are

$$(a) \quad \ddot{\theta}_m = \frac{T_m}{I_{\text{total}}} = \frac{8}{0.098} = 82 \text{ rad/sec}^2,$$

$$(b) \quad \ddot{\theta}_m = \frac{T_m}{I_{\text{total}}} = \frac{8}{0.0158} = 506 \text{ rad/sec}^2.$$

The no-load maximum angular acceleration of the motor would be about 530 rad/sec².

6.3 COMPARISON OF ACTUATING SYSTEMS

Table 6.1 is a summary of actuator characteristics that will be discussed later.

TABLE 6.1 SUMMARY OF ACTUATOR CHARACTERISTICS

Hydraulic	Electric	Pneumatic
<ul style="list-style-type: none"> + Good for large robots and heavy payload + 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 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 used 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 will 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 for pick and place + Compliant systems - Noisy systems - Require air pressure, filter, etc. - Difficult to control their linear position - Deform under load constantly - Very low stiffness. Inaccurate response - Lowest power to weight ratio

6.4 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. Many large robots of the past decade, mostly used in automobile production, were Cincinnati Milacron™ T3 hydraulic robots, or other brands with similar characteristics. The T3 robot offered a payload of over 220 lb. at 7 ft high: an impressive value. However, due to leakage problems, which is almost inevitable in hydraulic systems, and due to their power unit weight and cost, they are not used any more. Nowadays, most robots are electric. However, there are still many robots in industry that have hydraulic actuators. Additionally, for special applications such as very large robots and civil service robots, hydraulic actuators may be the appropriate choice.

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, where A is the ef-

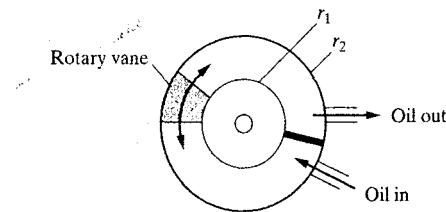


Figure 6.3 A rotary hydraulic actuator. This actuator can be directly attached to a revolute joint without any need for gear reduction.

effective area of the piston or ram and p is the working pressure. For example, for a pressure of 1,000 psi, every square inch of the cylinder develops 1,000-lb of force. In rotary cylinders, the same principle is true, except that the output is a torque where:

$$dA = t \cdot dr, \quad (6.4)$$

$$T = \int_{r_1}^{r_2} p \cdot r \cdot dA = \int_{r_1}^{r_2} p \cdot r \cdot t \cdot dr = pt \int_{r_1}^{r_2} r \cdot dr = \frac{1}{2} pt (r_2^2 - r_1^2),$$

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 diameters of the rotary cylinder, as shown in Figure 6.3.

The flow rate and volume of oil needed in a hydraulic system are

$$d(\text{Vol}) = \frac{\pi d^2}{4} dx, \quad (6.5)$$

$$Q = \frac{d(\text{Vol})}{dt} = \frac{\pi d^2}{4} \frac{dx}{dt} = \frac{\pi d^2}{4} \dot{x}, \quad (6.6)$$

where dx is the desired displacement and \dot{x} is the desired velocity of the piston. As you can see, by controlling the volume of the fluid going into the cylinder the total displacement can be controlled. By controlling the rate in which the fluid is sent to the cylinder, the velocity can be controlled. This is done through a servovalve that controls the servo valves later.

A hydraulic system generally consists of the following parts:

- Hydraulic linear or rotary cylinders and rams. These provide the force or torque needed to move the joints and are controlled by the servo valves or manual valves.
- A hydraulic pump which is a high-pressure pump that provides high-pressure fluid to the system.
- Electric (or others such as diesel engine) motor, which operates the hydraulic pump.
- Cooling system, which rids the system of the heat generated. In some systems, in addition to cooling fans, radiators and cooled air are used.
- Reservoir, which keeps the fluid supply available to the system. Since the pump is constantly supplying pressure to the system, whether or not the sys-

tem is using it, all the extra pressurized fluid, as well as all the returned fluid from the cylinders, flow back into the reservoir.

- Servovalve is a very sensitive valve that controls the amount and the rate of fluid to the cylinders. The servovalve is generally driven by a hydraulic servomotor.
- Safety check valves, holding valves, and other safety valves throughout the system.
- Connecting hoses, which are used to transport the pressurized fluid to the cylinders and back to the reservoir.
- Sensors, which are used to control the motion of the cylinders. They include position, velocity, magnetic, touch, and other sensors.

Figure 6.4 is a schematic drawing of a typical hydraulic system.

Figure 6.5 is a schematic drawing of a position control pilot valve for a hydraulic cylinder, also called a spool valve. This is a balanced valve, which means that the pressures on the two sides of the spool are equal. Thus, it takes very little force

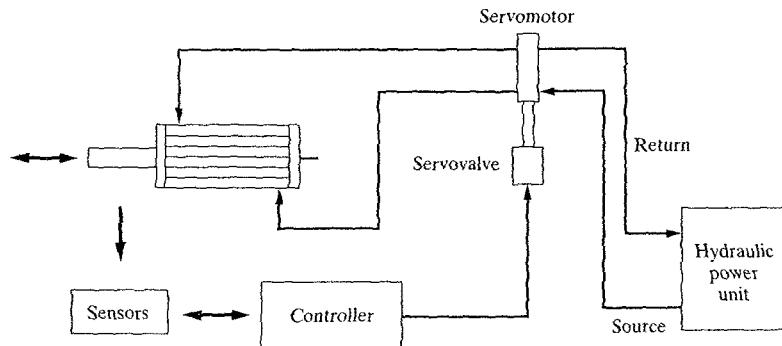


Figure 6.4 Schematic of a hydraulic system and its components.

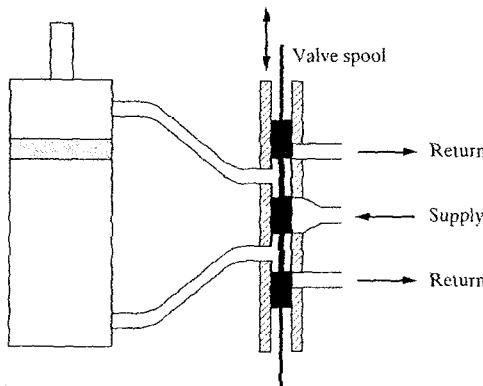


Figure 6.5 Schematic of a spool valve in neutral position.

to move the spool, even though it may be under very high pressures. When a servomotor is attached to the spool valve to operate it, a servovalve is created. The servo valve and the cylinder together form a hydraulic servomotor. As the spool moves up or down, it opens the supply and return ports through which the fluid travels to the cylinder or is returned to the reservoir. Depending on the size of the opening of the port, the supply fluid flow rate is controlled, and so is the velocity of the cylinder. Depending on the length of time that the port is kept open, the total amount of the fluid to the cylinder, and thus its total travel, is controlled. This can be written as

$$q = Cx \quad (6.7)$$

and

$$q(dt) = d(\text{vol}) = A(dy), \quad (6.8)$$

where q is the flow rate, C is a constant, x is the spool's displacement, A is the area of the piston, and y is piston's displacement. Combining Equations (6.7) and (6.8) and designating d/dt as D , we have

$$Cx(dt) = A(dy)$$

and

$$y = \frac{C}{AD} x, \quad (6.9)$$

which shows that the hydraulic servomotor is an integrator.

The command to the servomotor controlling the spool valve comes from the controller. The controller sets the current to the servomotor, as well as the duration the current is applied, which, in turn, controls the position of the spool. Thus, for a robot, when the controller has calculated how much and how fast a joint must move, it sets the current and its duration to the servomotor, which, in turn, controls the position and rate of movement of the spool valve, which, in turn, controls the flow of the fluid and its rate to the cylinder, which moves the joint. The sensors provide feedback to the controller for accurate and continued control. Figure 6.6 shows the flow of the fluid as the spool valve moves up or down. As you can see, a simple motion of the spool controls the motion of the cylinder.

To provide feedback to the servovalve, either electronic or mechanical feedback can be added to the valve. (Otherwise, it will not be a servovalve, but a manual spool valve). Figure 6.7 shows a simple mechanical feedback loop. Since this is a very simple system and can be easily understood, it is mentioned here. A similar design is used in a two-stage spool valve to provide feedback to the valve. As you can see in the figure, a simple lever is added between the output and the input to provide an error signal to the system. As the desired position for the load is set by the set-point lever, say, up, the spool valve is opened, which will operate the cylinder. However, in reality, this lever arm is providing an error signal to the system, and thus the system provides a signal (fluid pressure) to the cylinder. The error signal is integrated by the integrator (in this case, the cylinder), and as the error approaches zero, the signal to the system goes to zero. As the cylinder starts to move in the direction of the set desired position (in this case, upward), the error signal becomes increas-

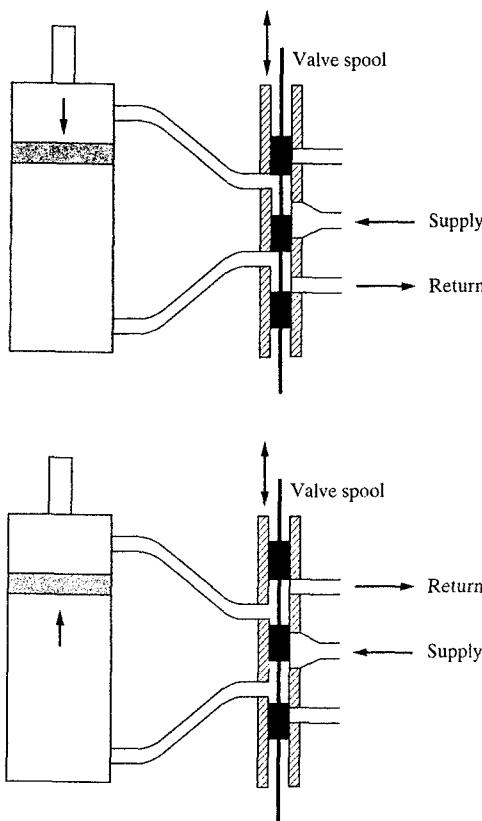


Figure 6.6 Schematic of a spool valve in open position. Depending on which ports are opened, the direction of motion of the piston will change.

ingly smaller, which, in turn, reduces the output signal by closing the spool valve. By the time the load is at the desired position, the spool valve is closed and the load stops. Figure 6.8 shows the schematic of the block diagram for this feedback loop.

As you can see, there are many intricate and small passageways inside a hydraulic valve — especially, in servovalves with feedback control integrated into the valve. In fact, this is why these systems are so susceptible to dirt or viscosity changes due to temperature. The smallest of foreign bodies can affect the servovalve by restricting its passageways or ports. Similarly, viscosity changes can change the response of the valve as the fluid becomes less or more viscous. To visualize how these valves are constructed, imagine that you slice the valve into thin layers. Each layer will have corresponding portions of the passageways and ports and openings in it. One may manufacture the slices, which is easy to do by different methods such as a press, assemble the layers, and then either diffuse the layers together into one piece (under proper pressure and temperature) or connect them into one piece by through-bolts. This method is used extensively in industry to manufacture valves

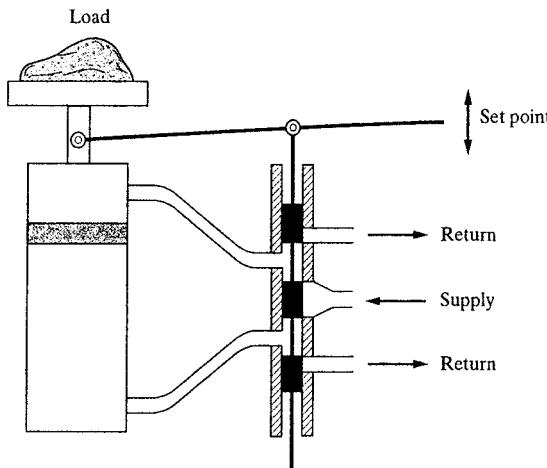


Figure 6.7 Schematic of a simple control device with proportional feedback.

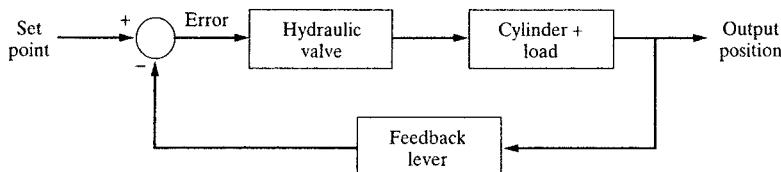


Figure 6.8 Block diagram of the hydraulic system with proportional feedback control scheme.

and many other products such as camera bodies, etc. Figure 6.9 shows a simple example of this technique. As you see, when the thin slices are put together, a three-dimensional object results. Many rapid prototyping machines use the same technique to create three-dimensional prototypes of products.

In this book, it is assumed that you have studied automatic controls in other courses and that you can understand how a servovalve may be controlled with PID or digital control schemes. This subject is not further discussed in this book.

Other designs have also been used for hydraulic actuation. For example, IBM 7565 robotic manufacturing system consists of a gantry, six-axis, hydraulic robot actuated at each of its three linear joints by a linear hydraulic motor. Each linear motor consists of a set of four small hydraulic cylinders, which sequentially move in and out against a waved surface, as shown in Figure 6.10. The four cylinders are forced to be in a position that corresponds to the desired position against the waved surface. This, in turn, forces the carriage to move sideways. The four cylinders are controlled by a single servovalve. The advantage of this system is that by adding simple sections of the waved surface, the actuation may be made as long as desired. Figure 6.11 shows this actuator on the IBM 7565 robot.

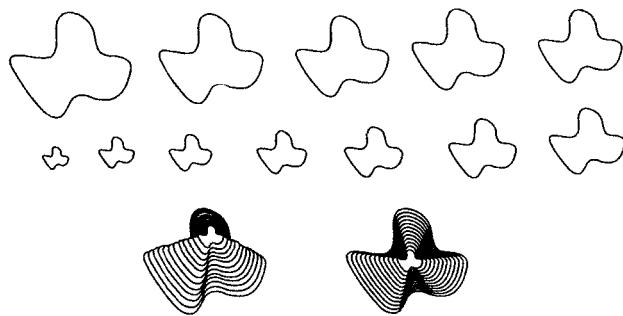


Figure 6.9 An object can be sliced into thin layers. When the layers are cut out individually and put together later they will recreate the object. This method is used for rapid prototyping and for manufacturing complicated parts with intricate internal openings and passageways.

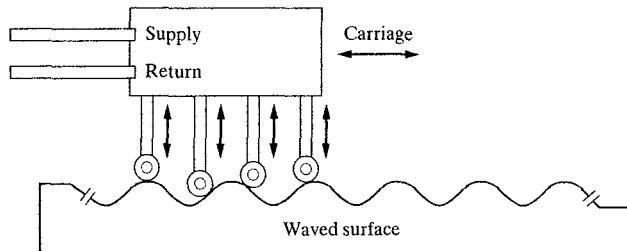


Figure 6.10 IBM 7565 linear hydraulic motor. As the four cylinders are moved in and out against the waved surface, the carriage moves sideways.

Another recent hydraulic actuator is modeled after biological muscles. A muscle actuates a bone by contracting and thus becoming shorter. It acts only in one direction, though, and as a result, needs to be used in opposing pairs. In the hydraulic version, a similar design is used, where an oval shaped bladder is placed inside a shear sheath, as shown in Figure 6.12. As the pressure in the bladder is increased, it becomes more spherical, causing the shear sheath to bulge out and become shorter, just like a muscle. This design is promising, but due to nonlinearities inherent in the system and also to technical difficulties, it has yet to become practical. However, since it looks like a biological muscle, it can be very useful in humanoid robots.

6.5 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 have lower inertial loads. How-

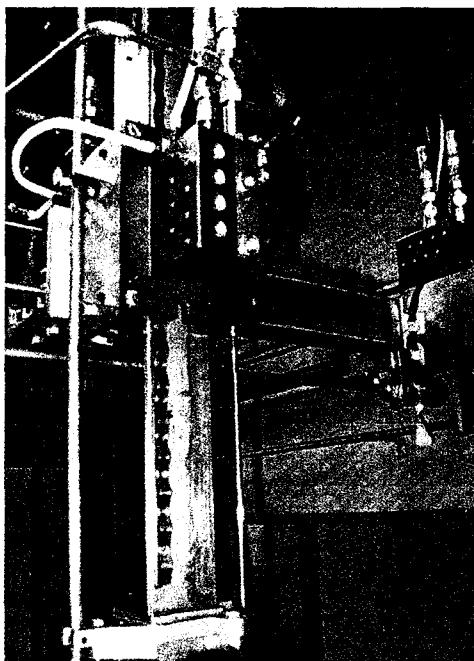


Figure 6.11 The linear hydraulic motor of IBM 7565 robot.

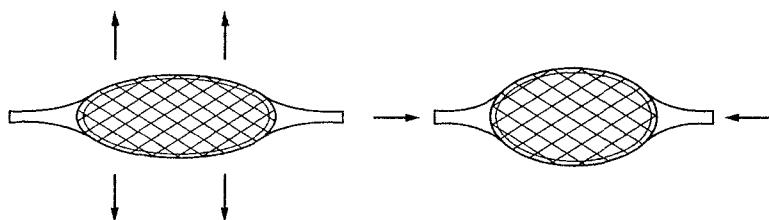


Figure 6.12 Schematic of a muscle-type hydraulic actuator.

ever, since pneumatic devices operate at a much lower air pressure, usually up to 100–120 psi, their power-to-weight ratio is much lower than hydraulic systems.

The major problem with pneumatic devices is that air is compressible, and thus it compresses and deforms under load. As a result, pneumatic cylinders are usually only used for insertion purposes, where the actuator is all the way forward or all the way backward, or they are used with 1/2-degree-of-freedom 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].

6.6 ELECTRIC MOTORS

When a wire carrying a current is placed within a magnetic field, it experiences a force normal to the plane formed by the magnetic field and the current as $\vec{F} = \vec{I} \times \vec{B}$. If the wire is attached to a center of rotation, the resulting torque will cause it to rotate about the center of rotation. Changing the direction of the magnetic field or the current causes the wire to continuously rotate about the center of rotation, as shown in Figure 6.13. In practice, to accomplish this change in the current, either a set of commutators and brushes are used for DC motors, the current is electronically switched for DC brushless motors, or AC current is used for AC motors [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 develops through the conductor. This is called a generator. Although we will discuss certain particular issues about electric motors, the assumption is that you have studied about different motors and how they operate in other courses. So, the discussion here will be at a minimum and only about the subjects that are directly related to robotic actuation.

There are many types of electric motors that are used in robotics. They include the following:

- DC motors
- reversible AC motors
- brushless DC motors
- stepper motors

Except for stepper motors, all other types of motors can be used as a servomotor, which 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. Some motors have permanent magnets (PMs). These motors generate less heat, since the field is always present and no current is needed to build them.

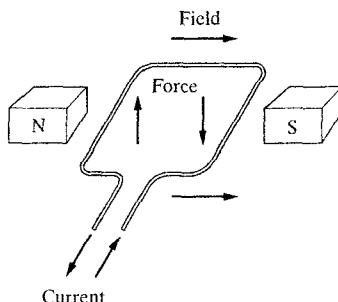


Figure 6.13 When a wire carrying a current is placed within a magnetic field it will experience a force in a direction normal to a plane formed by the current and the field.

Others have a soft iron core and windings, where an electric current creates the magnetic field. In this case, more heat is generated, but when needed, the magnetic field can be varied 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. For example, you should never take a motor apart, as the permanent magnet will become significantly weaker. This is because the iron mass around the magnet holds the field intact until they are separated. To increase the strength of the permanent magnets in motors, most manufacturers magnetize the magnets after assembling the motor. Motors without permanent magnets do not have this problem.

One important issue in the design and operation of all motors is the dissipation of heat. As with the heat generated in 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 well. The higher the current, the more heat is generated, as $W = RI^2$. Thicker wires generate less heat, but are more expensive, are heavier (more inertia), and require more space. All motors generate some heat. However, what is important is the path that the heat must take to leave the motor since if the heat is dissipated faster, more generated heat can be dissipated before damage occurs.

Figure 6.14 shows the heat leakage path to the environment for an AC-type motor and a DC-type motor. In DC-type motors, the rotor contains the winding and carries the current, and thus, heat is generated in the rotor. This heat must go 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 very good isolator. Thus, the total heat transfer coefficient for the DC motor is relatively low. On the other hand, in an AC-type motor, the rotor is a permanent magnet, and the winding is in the stator. The generated heat in the stator 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. As a result, AC-type motors can be exposed to relatively higher currents without damage, and thus they are generally more powerful for the same size. Stepper motors, although not AC motors, have a similar construction;

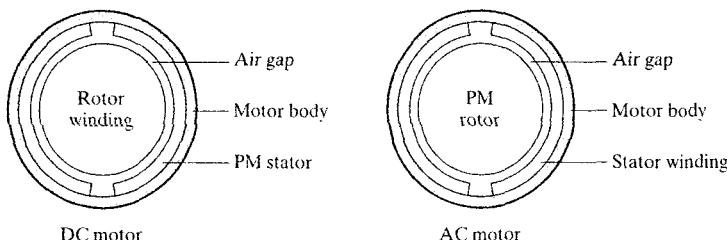


Figure 6.14 Heat dissipation path of motors.

the rotor is a permanent magnet, and the stator contains the windings. Thus, stepper motors have good heat dissipation capability.

Of course, another 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 thus they are sturdy and generally have long life (only limited by the life of rotor bearings).

In the sections that follow, each of the aforementioned motors will be discussed briefly.

6.6.1 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.

In DC motors, the stator is a set of fixed permanent magnets, creating a fixed magnetic field, while the rotor carries a current. Through brushes and commutators, the direction of current is changed continuously, causing the rotor to rotate continuously. Conversely, if the rotor is rotated within the magnetic field, a DC current will develop, and the motor will act as a generator. (The output is DC, but not constant.)

If permanent magnets are used to generate the magnetic field, the output torque T_M is proportional to the magnetic flux ϕ and the current in the rotor windings I_{rotor} . Then,

$$T_M = k_t \phi I_{\text{rotor}}, \quad (6.10)$$

where k_t is a constant. Since in permanent magnets, the flux is constant, the output torque becomes a function of I_{rotor} , 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:

$$T_M = k_t k_f I_{\text{rotor}} I_{\text{stator}}. \quad (6.11)$$

Here both k_t and k_f are constants.

Through the use of powerful magnets made of rare earth materials and alloys, the performance of motors has been improved significantly. 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 large size of many electric motors, a disk or shell motor can be used. In disk and shell motors, the iron core of the rotor winding is removed to reduce its weight and inertia, and as a result, these motors are capable of producing very large accelerations (zero to 2,000 rpm in one ms [3]); they respond very favorably to changing currents for control purposes. A shell motor's rotor looks similar to a regular DC rotor without the massive iron core. However, in a disk motor, the rotor is a flat, thin, plate, with windings pressed (etched) into it, as if one would flatten a rotor into a disk. The wires are generally cut out of a copper plate and embedded into a disk. The permanent magnets are

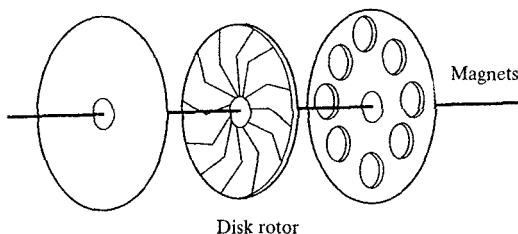


Figure 6.15 Schematic of a disk motor. The rotor has no iron core and thus has very little inertia. As a result, it can accelerate and decelerate very quickly.

generally small, short cylindrical magnets that are placed on the two sides 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. Figure 6.15 is a schematic of a disk motor.

6.6.2 AC Motors

Electric AC motors are similar to DC motors, except that the rotor is permanent magnet, the stator houses the windings, and all commutators and brushes are eliminated. This is possible because the changing flux is provided by the AC current, not by commutation. As the flux generated by the AC current changes, the rotor follows it and rotates. As a result, AC motors have fixed nominal speeds, a function of the number of poles on their rotor, and the line frequency (60 Hz). Since AC motors can dissipate heat more favorably than DC motors, they can be more powerful. The same principles of back-emf (please see section 6.6.5) hold for AC motors. There are also reversible AC motors available.

6.6.3 Brushless DC Motors

Brushless DC motors are a hybrid between AC motors and DC motors. Although not exactly the same, their construction is very similar to an AC motor. 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 a hall effect sensor, attached to the rotor, sends a signal to a controller, which switches the current to the rotor. For smooth operation and almost constant torque, the rotor usually has three phases in it [2,4]. Thus, three currents, with 120° phase shift, are fed into the rotor. Brushless DC motors are operated by a controller circuit. They will not operate if you connect them directly to a DC power source.

6.6.4 Direct-Drive Electric Motors

Direct-drive electric motors are very similar in construction to brushless DC motors or steppers. The major difference is that they are designed to deliver a very large

torque at very low speeds and with very high resolution. These motors are intended to be used directly with a joint without any gear reduction. Direct-drive motors are still very expensive and very heavy, but have impressive characteristics. In one model by NSK™ Corporation, a 40-Kg motor produces a continuous torque of 150 Nm at a maximum 3 rps, with a resolution of 30 arc-sec.

6.6.5 Servomotors

An important issue in all electric motors is the back electromotive force, or back-emf. As you remember, a wire carrying a current within a magnetic field will experience a force, which causes it to move. Similarly, if a wire (conductor) moves within a magnetic field such that it will cross the field lines, a current will be 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 in the magnetic field of the magnets, a current will be induced in them in the opposite direction of the input current. This current is called back-emf, and it tends to reduce the effective current of the motor. The faster the motor rotates, the larger the back-emf is. Back-emf current is usually expressed as a function of rotor speed:

$$V_{\text{emf}} = nK_E. \quad (6.12)$$

Here K_E is typically given in volts per 1,000 rpm. As the motor approaches its nominal no-load speed, the back-emf is large enough that the motor speed will stabilize 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

$$V_{\text{in}} = IR + V_{\text{emf}} = IR + nK_E, \quad (6.13)$$

where R is the windings' resistance. If a load is applied to the motor, it will slow down, resulting in smaller back-emf, larger effective current, and consequently, a positive net torque. The larger the load, the slower the motor will rotate in order to develop the 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. Unfortunately, in each case, when the back-emf is smaller, although 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.

To be able to develop larger torques without slowing down the motor, one would have to increase the current (to the rotor, the stator, or both if soft iron cores is used). In such a case, although the motor rotates at the same speed and the back-emf is still the same, the larger current will increase the net effective current and, thus, the torque. By varying the current (or corresponding voltage) the speed-torque balance can be maintained as desired. This system is called a servomotor.

The output torque T developed by the motor can be expressed as a function of the torque constant K_T , typically in oz.in/amp, as in $T = IK_T$. The opposing torques are friction torque T_f , viscous damping torque (as a function of speed) nK_D , and the opposing load T_L . Thus, the motor's torque is governed by

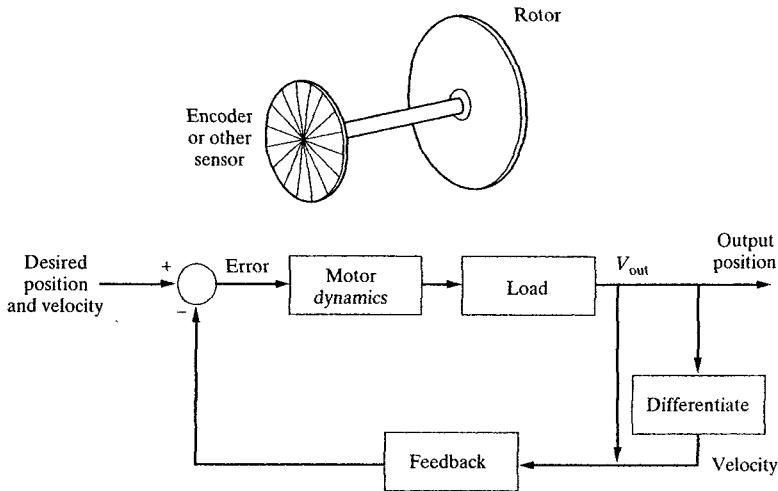


Figure 6.16 Schematic of a servomotor controller. A sensor sends velocity and position signals to the controller, which controls the output velocity and position of the servomotor.

$$T = T_L + T_f + nK_D = IK_T. \quad (6.14)$$

Equations (6.12), (6.13), and (6.14) govern the motions of the motor.

A servomotor is a DC, AC, brushless, or even stepper, motor with feedback that can be controlled to move at a desired speed (and consequently, torque), for a desired angle of rotation. To do this, a feedback device sends signals to the controller circuit of the servomotor reporting its angular position and velocity. If as a result of higher loads, the velocity is lower than desired set value, the current is increased until the speed is equal to the desired value. If the speed signal shows that the velocity is larger than desired, the current is reduced accordingly [5]. If position feedback is used as well, the position signal is used to shut off the motor as the rotor approaches the desired angular position.

We will discuss sensors later. For now, let it suffice 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 6.16 is a schematic of a simple control block diagram for a servomotor.

6.6.6 Stepper Motors

Stepper motors are versatile, long-lasting, simple motors that can be used in many applications. In most applications, the stepper motors are used without feedback. This is because unless a step is missed, a stepper motor steps a known angle each time it is moved. Thus, its 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, yielding it appropriate choice for certain applications. Most stepper motors can be used in different modes by wiring them differently.

Unlike regular DC or AC motors (but similar to 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 residual torque called detent torque. It requires an external torque to turn a stepper motor, even when not powered. As a result, all stepper motors need a microprocessor or driver/controller (indexer) circuit for rotation. You may either create your own driver or you may purchase a device called an indexer that will drive the stepper motor for you. As with servomotors, which need 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, stepper motors are hardly used, except in small table-top robots, or, in one case, an industrial robot with stepper motors and feedback. However, stepper motors are used extensively in nonindustrial robots and robotic devices, as well as in other devices that are used in conjunction with robots, from tooling machines to peripheral devices and from automatic manufacturing to control devices.

Structure of Stepper Motors Generally, stepper motors have permanent magnet rotors, while their stators house multiple windings. Based on the discussion in Section 6.6, since the heat generated in the coils can 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.

In different types of stepper motors, the permanent magnet rotors are different. 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 flux under the control of a controller or driver. In the next few sections, we will study how stepper motors operate.

Principle of Operation Imagine a stepper motor with two coils in its stator and a permanent magnet as its rotor, as shown in Figure 6.17. When each of the coils of the stator is energized, the permanently magnetized rotor (or soft iron in variable reluctance motors) will rotate to align itself with the stator magnetic field (a). The rotor will stay at this position unless the field rotates. As the power to the present coil is disrupted and is directed to the next coil, the rotor will rotate again to align itself with the field in the new position (b). Each rotation is equal to the step angle, which may vary from 180 degrees to as little as a fraction of a degree. (In this example, it is 90°). Next, the first coil will once again be turned on, but in the opposite polarity, while the second is turned off. This will cause the rotor to rotate 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 will bring the rotor back to exactly the same state it was at the beginning of the sequence. Now imagine that at the conclusion of the first step, instead of turning off one coil and turning on the second coil,

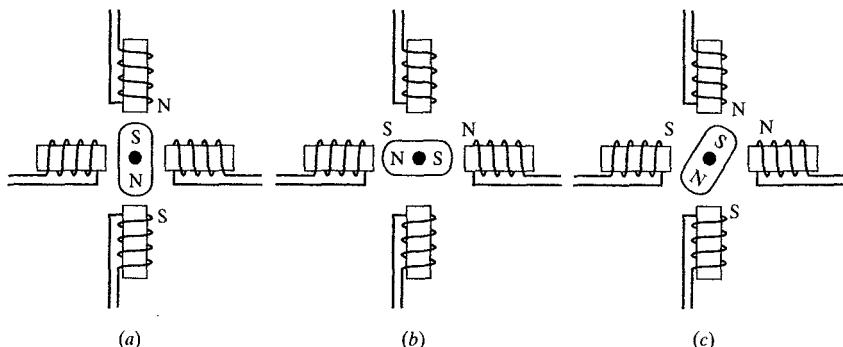


Figure 6.17 Basic principle of operation of a stepper motor. As the coils in the stator are turned on and off, the rotor will rotate to align itself with the magnetic field.

that both would be turned on. In that case, the rotor would only rotate 45° to align itself to the path of least reluctance (Figure 6.17(c)). Later, if the first coil is turned off while the second remains on, the rotor will rotate 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. Most industrial steppers run between 1.8 to 7.5 degrees at full stepping. Obviously, to reduce the size of the steps, the number of poles may be increased. However, there is a physical limit to how many poles 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, 50 teeth on the rotor and 40 teeth on the stator will result in a 1.8 -degree step angle with 200 steps per revolution, as will be discussed later.

Canstack Motors These motors are very common, are usually 7.5° steppers or similar, and are used in many different applications. Due to their construction, they are relatively flat and lend themselves to applications with low vertical clearance.

The rotor is a cylinder with alternate strips of north-south polarities, usually made of plastic embedded magnets, as shown in Figure 6.18. A typical rotor for a 7.5° stepper motor will have 24 poles on it.

The stator is made up of four shells, each with 12 teeth, stacked on top of each other, with the teeth staggered in 1–3–2–4 order, as shown in Figures 6.19 and 6.20. Each shell is a plate, with small poles around and perpendicular to it, as shown. One coil is wrapped around shells 1 and 2, as is one around shells 3 and 4. The four shells create two independent coils 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 will create a certain magnetic polarity opposite 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 will cause all the teeth on each shell of a pair to be of similar polarity—all North or all South. Conse-

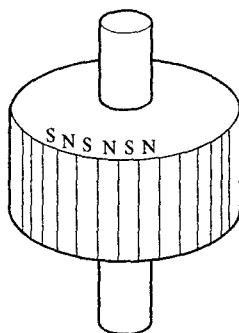


Figure 6.18 Schematic of a canstack rotor.

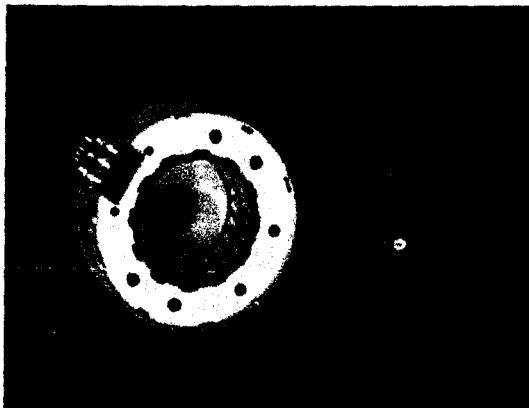


Figure 6.19 Canstack stator and rotor.

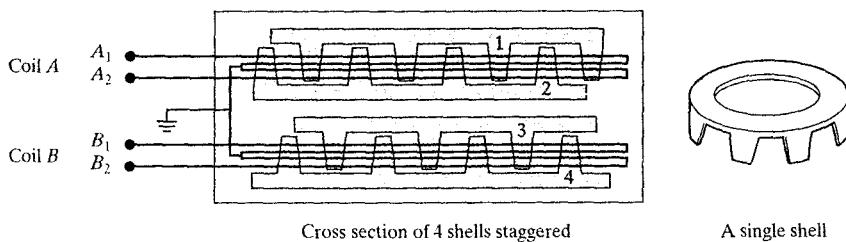


Figure 6.20 Canstack stator windings and shells.

quently, energizing both windings will create repeating patterns of North and South in all four shells, depending on the polarities.

Figure 6.20 depicts a schematic of a canstack stator, linearized for better visualization. Shells 1 and 2 are related to coil A1/A2 (two halves of coil A), as are shells 3 and 4 to coil B1/B2 (two halves of coil B). If coil A1 is turned on, shell 1 will be North and shell 2 will be South. However, if coil A2 is turned on, the polarity of

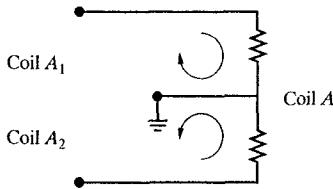


Figure 6.21 Center tapping of a coil allows changing the polarity of the magnetic field by having the current flow in either half of the coil.

shells 1 and 2 will be reversed; shell 1 will be South and shell 2 will be North. The same will happen to shells 3 and 4 with coils B1 and B2. This is called center tapping and is shown in Figure 6.21. The advantage of center tapping is that during manufacturing, a set of two wires are simultaneously wrapped around the pole. At the conclusion, two heads of the wires are connected and form the ground, while the other two heads will be contact points for input current. As a result, the pole 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 will be turned on in the following sequence, resulting in the indicated polarities:

Step	A1	A2	B1	B2	Shell1	Shell2	Shell3	Shell4
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 together (as they would cancel each other's fields). During each step of the sequence, the poles on the rotor will align themselves between the stator poles 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 6.22. In this figure, "arcs" of south–north 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. Thus, 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. Please note that Figure 6.22 is drawn with only 1/4 of all stator and rotor poles. In actual stepper motors, there are a total of 48 poles, providing a 7.5° step angle.

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, thus half-stepping. However, this is not common in canstack motors. Table 6.2 shows the on–off sequence for full stepping, as well as for half-stepping the, stepper motor.

A simple C program to drive a stepper motor with the Mini-Board microprocessor is presented next. Although some of the commands are specific to Mini-

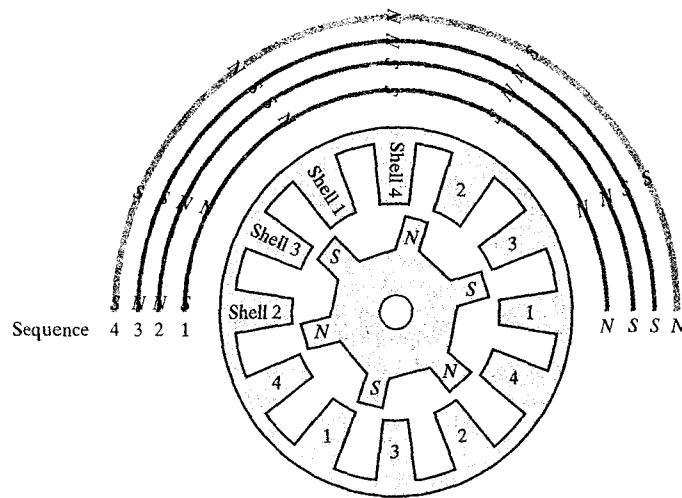


Figure 6.22 The cross section of a canstack stepper motor. Each south-north arc shows the polarities of each pole during one step of the sequence of four steps.

TABLE 6.2 FULL STEP (A) AND HALF-STEP (B) SEQUENCE FOR STEPPER MOTORS. REVERSING THE SEQUENCE CAUSES THE STEPPER MOTOR TO ROTATE IN THE OPPOSITE DIRECTION

(a)				(b)					
Step	A1	A2	B1	B2	Step	A1	A2	B1	B2
1	On	Off	On	Off	1	On	Off	On	Off
2	Off	On	On	Off	2	Off	Off	On	Off
3	Off	On	Off	On	3	Off	On	On	Off
4	On	Off	Off	On	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

Board, similar programs may be written for other microprocessors that work with a C compiler. Microprocessors that work with assembly language require assembly programs specific to their design. In this program, four output ports of the processor are sequentially turned on and off according to Table 6.2. Motor(n,15) turns on output port # n at full pulse width modulated voltage. Motor(n,0) turns off port # n.

/* Driving a Stepper Motor

*/

/* This program runs a stepper motor that is connected to all four

*/

```

/* motor output ports of the Mini-Board. It will rotate the motor for */
/* the specified number of steps in one direction, then in the */
/* opposite direction. */

#define WANT_MOTORS
#include <mboard.h>
#include <mbintsvc.h>

main ()
{
    int a, steps;

    motor(1,15); motor(4,15); /* Initialization */ */

    steps = 50; /* Number of steps x 4 */
    a=9; /* Pause value for maximum speed control */ */

    { while ( steps >= 0 )

        { motor(1,0); /* Turn off output port 1 */ */
          motor(2,15); /* Turn on output port 2 */ */
          msleep(a);

          motor(4,0); /* Turn off output port 4 */ */
          motor(3,15); /* Turn on output port 3 */ */
          msleep(a);

          motor(2,0); /* Turn off output port 2 */ */
          motor(1,15); /* Turn on output port 1 */ */
          msleep(a);

          motor(3,0); /* Turn off output port 3 */ */
          motor(4,15); /* Turn on output port 4 */ */
          msleep(a);

          steps = steps—1; }
      }

    steps = 50; /* Rotation in the opposite direction */ */

    { while ( steps >= 0 )

        { motor(3,0);
          motor(4,15);
          msleep(a);

          motor(2,0);
          motor(1,15);
          msleep(a);
    
```

```

motor(4,0);
motor(3,15);
msleep(a);

motor(1,0);
motor(2,15);
msleep(a);

steps = steps—1; }}

while(1)

{
    motor(2,0);
    motor(3,0); } /* Turn off all ports. */
*/
```

Hybrid Stepper Motors These steppers usually are 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, so that one end of the rotor is north and the other end is south (Figures 6.23 and 6.24). The rotor cylinders and the stator poles are all cut to have teeth, where the teeth on one half of the rotor is 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 will review the concept behind a caliper.

Let's take two parallel lines, capable of sliding relative to each other, each one unit of length long. Next, divide each line to 10 equal divisions. To have the division lines aligned with each other when moving one step, the sliding line will have to move one whole division to the next line (Figure 6.25).

Similarly, take two other lines, divide one into 10 equal divisions and the other line into 11 equal divisions. In this case, the divisions will be 0.1 and approximately 0.09, 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

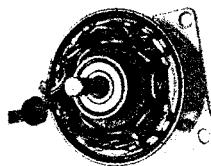


Figure 6.23 Hybrid stepper motor.

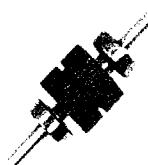


Figure 6.24 Rotor of a hybrid stepper motor.

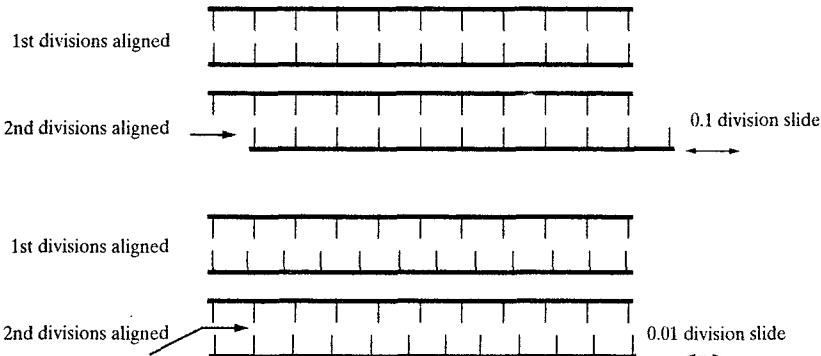


Figure 6.25 Application of unequal divisions for measuring lengths as in a caliper.

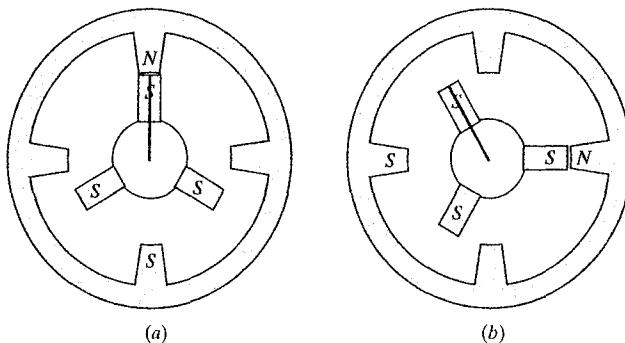


Figure 6.26 Basic operation of a hybrid stepper motor.

lines. Calipers do exactly the same. By dividing the lines into different lengths, one can easily measure a distance at a fraction of the smallest division.

The teeth on the stepper motor stator and rotor are also the same. 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 will be 1.8 degrees at full steps, because

$$\frac{360^\circ}{40} - \frac{360^\circ}{50} = 9^\circ - 7.2^\circ = 1.8^\circ. \quad (6.15)$$

The two rotor cylinders (Figure 6.24) are a half-step out of phase, and the stator poles run across the whole length of the stepper. Although the construction of hybrid and canstack steppers are different, driving the motors is similar.

Figure 6.26 is a simplified hybrid stepper motor with only two coils and a rotor with three lobes (teeth). As was mentioned earlier, in actual motors, many teeth are cut into the poles to create smaller steps, but to see how the motor works, this example only uses three teeth. As long as the number of teeth in the stator and the

rotor are different, the aforementioned effect will be true. Now suppose that one coil is energized as shown in (a). 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 in the figure. 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 (b). The same sequence will continue as was mentioned for canstack motors, and the rotor will rotate. Changing the sequence backwards will cause the rotor to rotate backwards. Applying the sequence faster will cause the rotor to move faster. Thus, by controlling the sequence, its direction, and its speed, we can control the rotor's motion and angular velocity. 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 powering 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 will develop its full power.

In bipolar motors, the assumption is that the polarity of the power source can be changed. As a result, there will either be 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 not damage the circuits. In this case, the motor will require more power supplies or electronics, but can be run either at full or half-step modes and will develop 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 flowing the current in each half of the coil. Thus, with simple circuits and one power supply, the motor can be run in either full or half-step mode, but since only half the coil is energized, the motor only develops half of its full power. Figure 6.27 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 microcomputer.

Most motors are wired to be used in any of the three modes. Some motors are not. Figure 6.28 shows the schematic of different possibilities of coil wiring combinations in motors. In eight-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 six-lead motors are connected such that each coil is center tapped, but the two coils are detached. In five-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, one 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, as it is assumed that stepper motors advance a known angle every time a signal is sent. As long as the

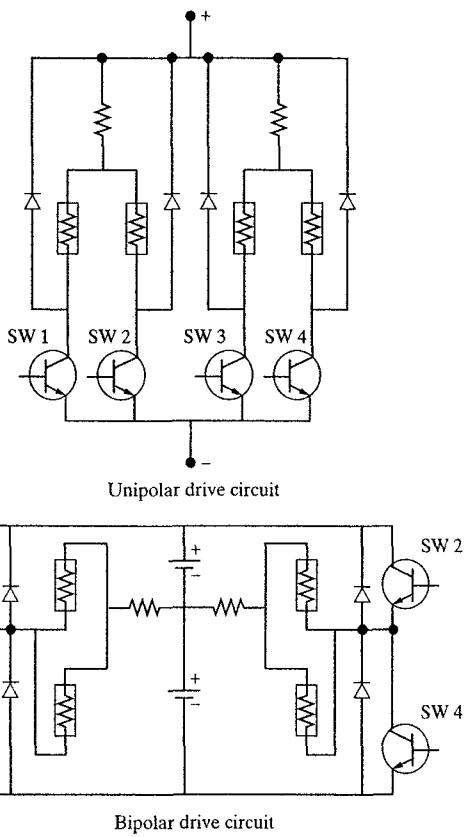


Figure 6.27 Schematic drawing for unipolar and bipolar drive circuits.

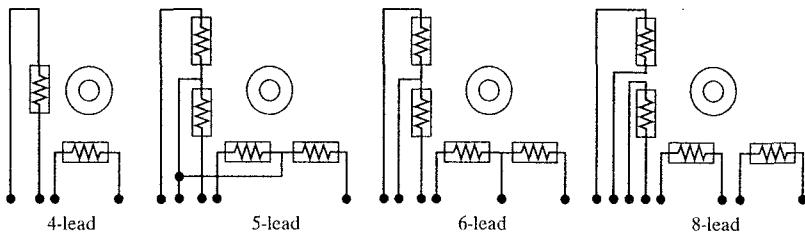


Figure 6.28 Stepper motor lead configurations.

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, in fact, 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 wrong.

Stepper motors develop their maximum torque, called holding torque, at zero angular velocity, when the rotor is stationary. (The torque developed with no power

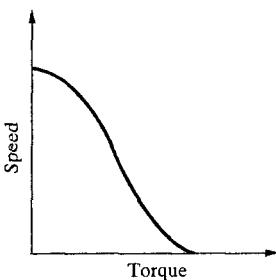


Figure 6.29 A typical speed–torque curve.

is called detent torque.) As the speed of the motor increases, the torque it develops reduces significantly. Thus, it is very important that the designer check the speed–torque characteristic of stepper motors from manufacturers before a choice is made. Figure 6.29 is a typical speed–torque characteristic.

One reason for this poor performance at higher speeds is that since at each step, the rotor must first accelerate, cruise, and then decelerate and stop at the next step, and that this process must be repeated for every single step, stepper motors cannot rotate fast. If the signals coming to the motor are too fast, the rotor will not have the time to accelerate/decelerate and will miss steps. So, one reason is the inertia of the rotor. However, even more important is the alternating magnetic fields of the stator. As was discussed earlier, every time a coil is turned off, there will be a changing (in this case, a decaying) flux. The wires of the coils, in the presence of a changing flux, will develop a back-emf that will slow down the decay of the flux as a new one builds up due to the back-emf current. The generated flux will tend to “hold” the rotor from rotating and will slow it down. As a result, the rotor is pulled back and will not be able to rotate freely.

To remedy this problem and to prevent high-current “sparking” across the switches, a freewheeling diode may be added to the circuit, as in Figure 6.30(b). The diode will allow 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 6.30(c). 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.

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, usually up to 250 steps. For example, suppose that coil *A* is on and coil *B* is off. In full stepping, the next sequence will be for coil *A* to be off and for coil *B* to be on. In microstepping, this is divided into, say, 100 divisions. As a result, in the next microstep, coil *A* will be 99% on and coil *B* will be 1% on. As a result, 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* would become 98% strong and coil *B* would be at 2% level, 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 would divide

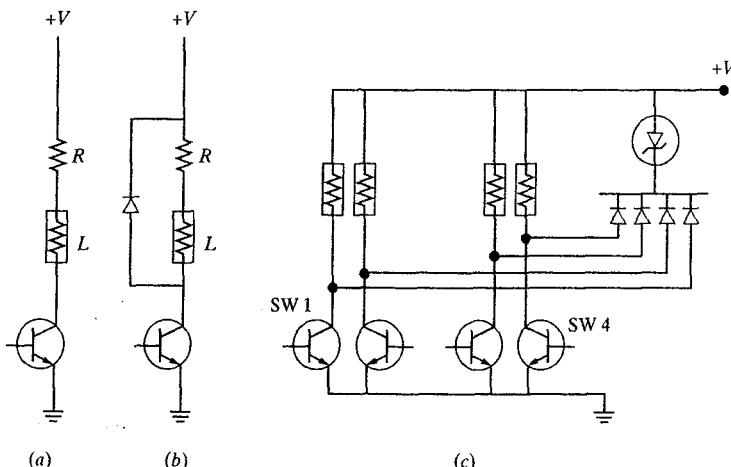


Figure 6.30 Application of (a) a resistor, (b) a diode, and (c) a zener diode with stepper motors for increasing the maximum velocity.

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, stepper motors' performance is improved significantly. They develop higher torque, and their maximum speed without missed steps is much higher. Additionally, because microsteps are very small, the motor's movement is not as piecewise, and thus 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.

In practice, instead of dividing the steps into linear and equally divided divisions (such as 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. Then, principally, in this mode, the stepper is in fact similar to an AC synchronous motor, except that its angular velocity is not fixed with 60-Hz line frequency, but is driven by the microstepper driver circuit. As a result, the synchronous motor is driven at any 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 dedicated stepper driver/indexers that motor manufacturers provide.

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

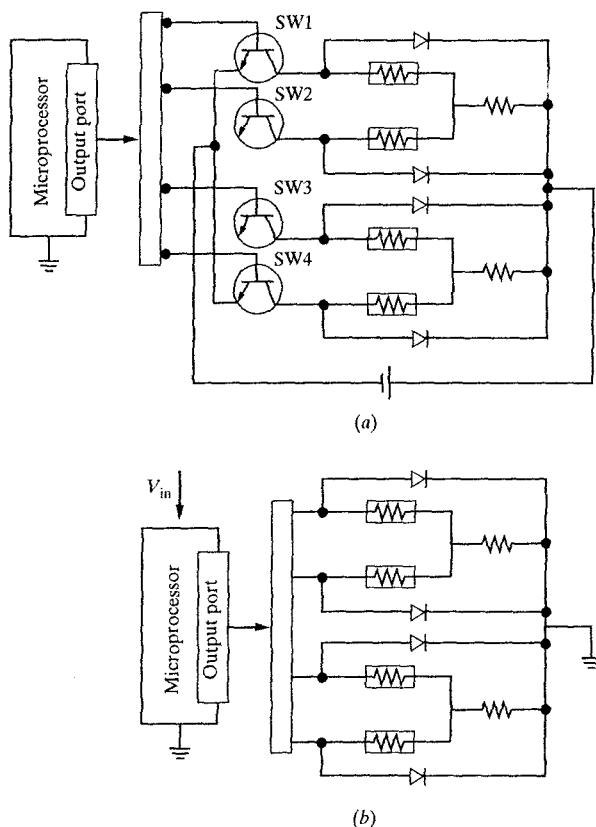


Figure 6.31 Schematic drawing of the application of a microcontroller in driving a stepper motor (a) with low current output, and (b) with high current output.

power (mA), it will not be able to provide high enough current to the motor windings. (The RS-232 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 (such as in Mini-Board [6], in which, the output (motor) port is pulse-width modulated and can deliver close to 1 amp), so long as the current requirement of the motor is below the level of 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 previously discussed. 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 6.31 shows a schematic arrangement in which a microprocessor with a low

power output port is used to drive a stepper motor with transistors (a), as well as a microprocessor with high current capability, which is used without transistors (b).

Alternatively, it is possible to use a dedicated integrated-circuit chip to control the motion of a stepper motor [7,8]. These IC chips, called stepper drivers or translators, are designed to sequence the stepper motor based on the information they receive. In most cases, the information needed is a pulse. Every time the translator receives a pulse (i.e., when the input to the chip changes from low to high), it will sequence the stepper by one step. If n signals are received, the motor will be sequenced for n steps. When a translator is used, the microprocessor only provides the pulses, which are always the same, and not the sequences. A second input pulse to the translator determines the direction of motion. Consequently, only two output ports are needed to drive a stepper motor for any displacement, at either direction, and at any speed. Thus, using stepper translators, the same number of output ports as before (four outputs) can drive twice as many stepper motors. It should be mentioned that most translators provide a choice of full or half stepping by making one of their pins high or low. In this case, three output lines will be necessary. Translators are very easy to use, are very inexpensive, and simplify programming of the processors. Translators, such as microprocessors, may be low current or high current. If they are low current (such as Motorola MC3479 with 0.35 amps and SGS-L297), it is necessary to use power transistors (such as SGS-L298) as switches, where the translator turns the transistors on and off, thus driving the stepper motor. If the translator is high current (such as Allegro MicroSystems Inc. 5804, SLA7024M, SLA7026M, SMA7029M, SLA7042M, and SLA7044M with current ratings from 1 to 3 amps), the output port of the translator can be connected directly to the stepper motor, simplifying the circuit. Figure 6.32 shows a schematic of a typical stepper motor translator (Allegro Micro Systems 5804 chip). Figure 6.33 shows how it can be used. As shown, the microprocessor provides a pulse train to pin 11 of the translator, which automatically sequences the stepper motor for each pulse. The faster the pulses, the faster the stepper motor's rate of rotation becomes. To control the direction of rotation, pin 14 of the translator must change from high to low or from low to high.

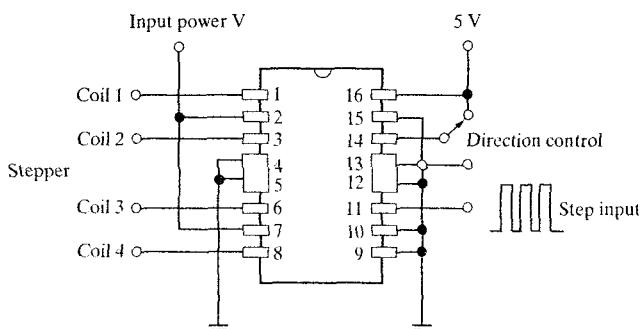


Figure 6.32 Schematic of a typical stepper motor translator.

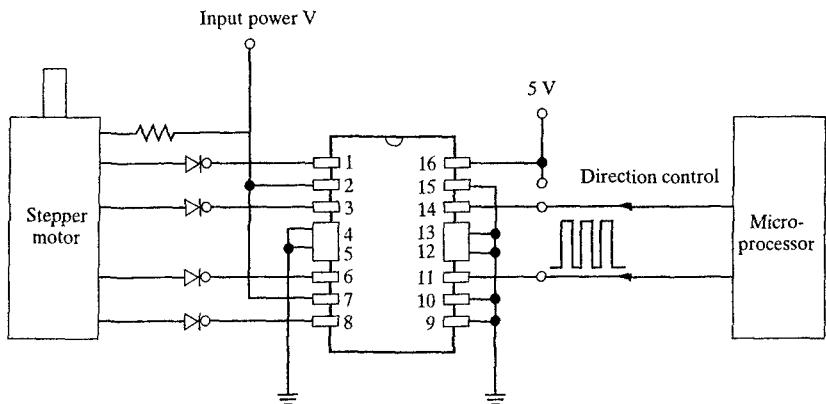


Figure 6.33 Schematic drawing of the application of a translator to run a stepper motor with a microprocessor.

The C program presented next is written for a Mini-Board microprocessor to run a stepper motor with a driver. Although some of the commands used are specific to the Mini-Board, similar programs may be written for other microprocessors that run with a C compiler. Microprocessors that run assembly language programs require assembly programs.

```
#define WANT_MOTORS
#include <mboard.h>
#include <mbintsvc.h>

void
main ()
{
    int a, step;          /* variable declaration */

    step=400;             /* Number of steps the motor moves */
    a=3;                  /* This value is used with the msleep
                           /* command to control speed. */

    while (step>=0)
    {
        motor(1,15);      /* This command turns on output port #1 at
                           /* maximum level of 15, creating a high. */
        msleep(a);         /* Execution pause of "a" msec. */
        off(1);            /* Output port #1 is turned off, creating a
                           /* low, and thus, an impulse. */
        msleep(a);
        step=step-1;
    }
}
```

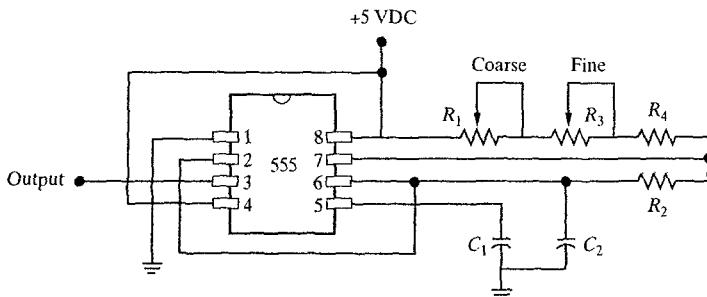


Figure 6.34 Schematic drawing of a timer circuit that creates an impulse train, which can be used to drive a stepper motor indexer. Adjusting the potentiometers will change the period of the output impulses.

This program creates a simple train of impulses that operate the stepper indexer. Within the program, a second port may be turned on or off to change the direction of the motor.

Another way to run a stepper motor with a translator is to provide the impulse train to the translator by means other than a microprocessor. For example, the timer circuit in Figure 6.34 can provide a steady stream of impulses to the translator, 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 will be needed (e.g., by 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. For $R_H = R_1 + R_3 + R_4$, the following equations apply to this circuit:

$$\begin{aligned} t_{\text{high}} &= 0.693 C_2 (R_H + R_2), \\ t_{\text{low}} &= 0.693 C_2 (R_2), \\ t_{\text{total}} &= 0.693 C_2 (R_H + 2R_2). \end{aligned} \quad (6.16)$$

For practical purposes, $\text{Max}(R_H + R_2) \leq 3.3 \text{ M}\Omega$, $\text{Min}(R_H) = 1 \text{ K}\Omega$, and $500 \text{ pF} \leq C_2 \leq 10 \mu\text{F}$. With 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 [9,10,11,12,13].

6.7 MICROPROCESSOR CONTROL OF ELECTRIC MOTORS

As was mentioned in Chapter 1, a robot is supposed to be a manipulator that is controlled by computers or microprocessors. So, of course, it is important to be able to control the motions of the electric motors 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 sections that follow, a few other common techniques for controlling electric motors will be mentioned.

A microprocessor is a digital device. It only deals with digital inputs and digital outputs. Any voltage that is lower than about 0.8 volt is considered to be low (off, 0). Any voltage greater than 2.4 volts is considered to be 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 (ADC) and digital-to-analog converters (DAC) are used for this purpose. However, a key element in both DAC and ADC's 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. To add to this capability, one may use two bits. Then, there are four possibilities, 00, 01, 10, 11. As the number of bits increases, the variations increase with 2^n . Thus, a four-bit set will have 16 distinct possibilities, and an eight-bit set would have 2^8 , or 256, possibilities. Every four bits is called a nibble, and every eight bits is a byte.

Suppose that you want to read in 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. Thus, the voltage can be divided into four different values — perhaps 0, 1.67, 3.34, and 5, which corresponds to 00, 01, 10, 11 bit mapping. Then we will be able to distinguish four different levels of voltage with the processor. If this resolution is not enough, one would have to increase number of bits. With a four-bit input, the 5-volt voltage can be divided into 16 portions, corresponding to 0000, 0001, 0010, 0011, ..., mapping. Then the smallest value of the voltage we could read is 0.33 volt. As you see, the larger the number of bits, the better the resolution becomes. However, what this means is that in order to read one input voltage, four input ports of the processor would 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 that you want to control a servomotor with a microprocessor. 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. Now you can imagine how many input and output ports would be necessary to have an accurate six-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 would have to calculate 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, alternatively, determines how much and how fast the servomotors of each

joint must rotate (based on the gear ratio of the joint, as well as other information). This information is converted to a set of voltages and voltage profiles that govern the servomotors. This means that in order to have a particular servomotor rotate at a particular speed, it will require a particular voltage. The processor calculates this voltage, 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.

6.7.1 Pulse Width Modulation

As was previously mentioned, to be able to run a servomotor with almost continuous voltage, one would have to dedicate a large number of output ports or bits to it to be able to have a good resolution. This is expensive, and at times when large number of inputs and outputs are needed, it is prohibitive. Instead, other means of creating a variable output can be used that require far fewer bits of information and are much more efficient for microprocessors. One such technique is called pulse width modulation (PWM).

PWM is used for DC motor speed control with microprocessors. Commonly, to control the speed of a DC motor, the applied voltage is changed; as the voltage increases or decreases, the speed of the motor is changed accordingly. However, as was already mentioned, when the speed is to be controlled by a microprocessor, changing the voltages to servomotors requires many bits of information. Alternatively, using PWM, many voltage levels can be achieved with one single-level input voltage, namely, the high voltage (say, 5 volts) and one output bit. To do this, the voltage on the output port of the processor is turned on and off repeatedly, many times a second, so that by varying the length of time that the voltage is on or off, the average effective voltage will vary. In other words, as shown in Figure 6.35, as t_1 vs. t changes, the average voltage at the motor changes accordingly. Since the rate of

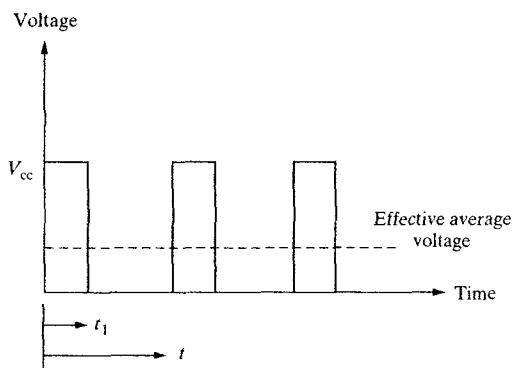


Figure 6.35 PWM timing.

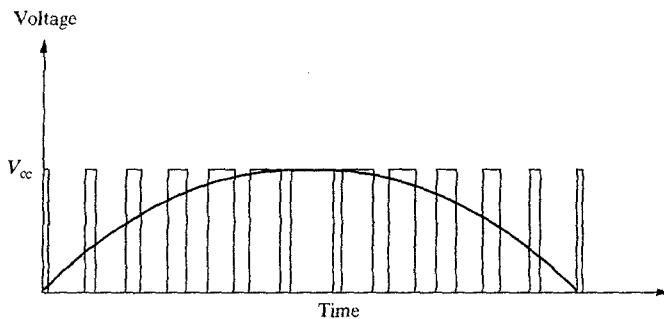


Figure 6.36 Sine wave generation with PWM.

on-off switching at the port is many times faster than the natural frequency of the motor's rotor, the switching will have little effect on the performance of the motor. The average output voltage in PWM is

$$V_{\text{out}} = V_{\text{cc}} \frac{t_1}{t}. \quad (6.17)$$

Similarly, by varying the timing continuously, a variable voltage can be created which can be used in brushless DC motors or similar applications. Figure 6.36 depicts a sine wave generation with PWM. Typical commercial PWM chips such as Allegro MicroSystem, Inc., 2916, 2998, 3952, and L6219DS are readily available for use.

6.7.2 Direction Control of DC Motors with an H-Bridge

Another troublesome issue in control with microprocessors is change of polarity for directional change. In microprocessor control of motors, it is desirable to change the direction of current flow in a motor for changing its direction of rotation with only two bits of information. In other words, instead of actually changing the polarity, one should change the direction of the flow by changing bit information from the microprocessor. A simple circuit called H-bridge can accomplish this as shown in Figure 6.37. As the two output bits of the processor change from high to low or vice versa, the direction of current flow in the motor will change as well, without requiring any polarity change. When transistors SW1 and SW4 are high, the current flows from A to B. When SW2 and SW3 are high, the current flows from B to A, thus changing the polarity and the direction of rotation of the motor.

6.8 MAGNETOSTRICTIVE ACTUATORS

When a piece of a material called Terfenol-D is placed near a magnet, this special rare-earth-iron material will change its shape slightly. This phenomenon is called

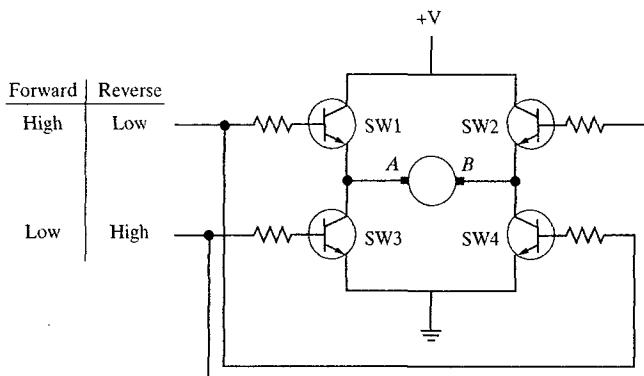


Figure 6.37 Application of H-bridge for motor direction control.

magnetostriction effect and is used to make linear motors with μ -inch displacement capabilities [14,15]. 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 one. A similar concept is used with piezo crystals as well to create a linear motor with nanoinch displacements [16].

6.9 SHAPE-MEMORY-TYPE METALS

One particular type of shape memory alloy, called BiometalTM, a patented alloy, shortens 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 thus shortens. However, unlike many other shape-memory alloys, it will once again switch back to martensitic state when it is cooled. 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 may come from an electric current flowing through the metal itself, which warms, 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, and thus, except in on-off situations, it is very difficult to accurately control the strain, and thus, the displacement [17].

Although Biometal wires are not yet fit to act as actuators, based on past experiences, it is possible to expect that someday they will become useful. In that case, a robot arm could be developed with muscles similar to that of humans or animals, which would only require a current to operate, as muscles do. Figure 6.38 shows a three-fingered end effector with Biometal wires as actuators.

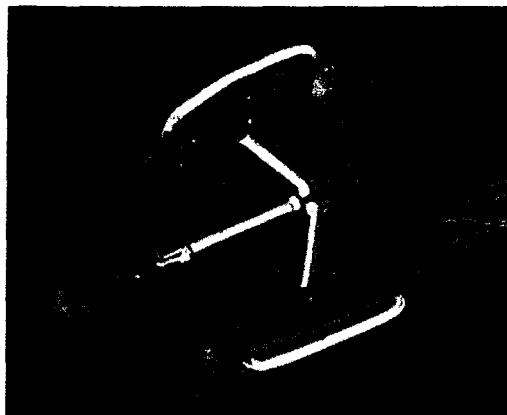


Figure 6.38 A three-fingered end effector with Biometal-wire actuators.

6.10 SPEED REDUCTION

Any means of speed reduction common in industry may be used in robotics 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 subsequent 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 for a gear train with corresponding angular velocities and number of teeth for the first and last gears:

$$\frac{\omega_{L/A}}{\omega_{F/A}} = \frac{\omega_L - \omega_A}{\omega_F - \omega_A} = \frac{N_F \times N_3}{N_2 \times N_L}, \quad (6.18)$$

where $\omega_F, \omega_L, \omega_A$ are the angular velocities of the first and last gears and of the arm, N_L, N_F, N_2 , and N_3 are the number of teeth of the first, last, and the planet gears, as shown in Figure 6.39.

In this gear-train example, it is assumed that the first gear is stationary and that 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 Equation (6.18), 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

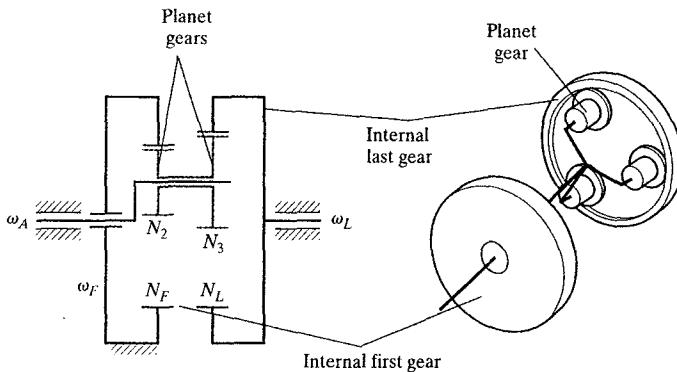


Figure 6.39 Schematic drawing of a planetary gear train.

$$\frac{\omega_L}{\omega_A} = \frac{N_2 N_L - N_F N_3}{N_2 N_L}, \quad (6.19)$$

which 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 so as to render very large gear ratios with very few gears. To do this, suppose that the teeth are chosen such that $N_F = N_2 + 1$ and $N_L = N_3 + 1$. In that case, Equation (6.19) reduces to

$$\frac{\omega_A}{\omega_L} = \frac{(N_F - 1)N_L}{N_F - N_L}. \quad (6.20)$$

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 one 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 teeth like this will not rotate relative to each other. To overcome this problem, the planets in Harmonic Drive are flexible metal bands with teeth, where they rotate with the aid of a cam (Figure 6.40). As the cam rotates, it deforms the band and will allow it to remain engaged, but rotate relative to the first and last gears while still having the intended number of teeth. This is schematically shown in Figure 6.41. For more information, please refer to manufacturer catalogs [19,20,21].

The nutating gear train concept is somewhat similar. Gears that are very close in size wobble relative to each other at their edges [22,23]. As a result, after each revolution, one gear falls slightly behind the other, thus creating a large gear ratio. The equations governing the systems are the same as Equation (6.19). A simple rendition of the nutating gears is shown in Figure 6.42.

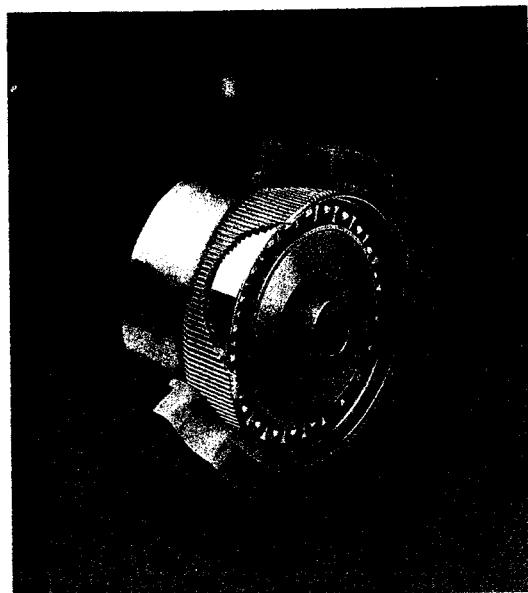


Figure 6.40 A Harmonic Drive.
Reprinted with permission from Harmonic drive Technologies.

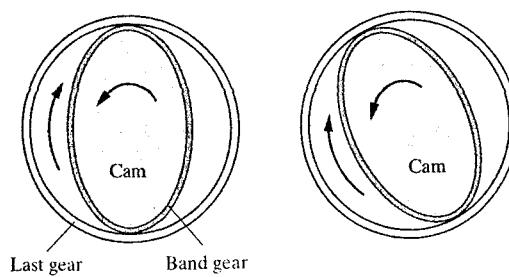


Figure 6.41 Schematic of the Harmonic Drive gear train.

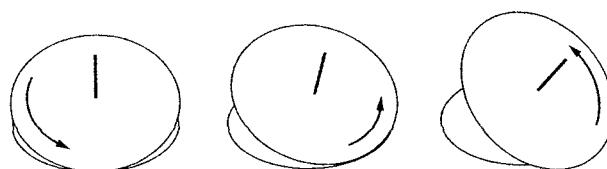


Figure 6.42 Nutating gears trains.

6.11 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, etc. Additionally, make sure that the actuator you pick can deliver the desired amount of 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 translator chips, commercial stepper drivers, or 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 servocontroller. Commercial servocontrollers are expensive. However, with an appreciable amount of effort, one may design and build a servocontroller. Please note that it is relatively easy to make a controller for controlling position of a servomotor. This can be done with a simple feedback device such as a potentiometer or an encoder. However, the design of a servocontroller becomes much more complicated if you decide to control the velocity of the robot as well (which is why a servomotor would be used).

You must also pick an appropriate microprocessor with enough computing power to calculate the kinematic equations, but also 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 programming the robot, it should function properly. In the next few chapters, we will learn more about sensors, at which time, we will integrate desired sensory information into the robot as well.

6.12 DESIGN PROJECT 2

This project involves the design and manufacturing of a rolling-cylinder robot rover. It is meant to consist of two colinear hollow cylinders that can rotate independently and thus navigate. Its power source, the electronic drive circuitry, the actuators, and the sensors are supposed to all be inside the two cylinders. As a result, from outside, the rover should look like a cylinder. However, you should be able to program the rover to follow a predetermined path, or using its sensors, navigate through hallways, a maze, or similar environments. Figure 6.43 is a schematic depiction of the design. As shown, if the two cylinders rotate relative to each other, the rover will rotate about a vertical axis. If they rotate together in the same direction, the rover will move forward. The exploded view shows how the circuitry and the drive motors may be mounted on a platform and assembled inside the cylinders. The two large cylinders are in fact two tires attached to the two motors. (Stepper motors are the best choice for this purpose.) 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 as-

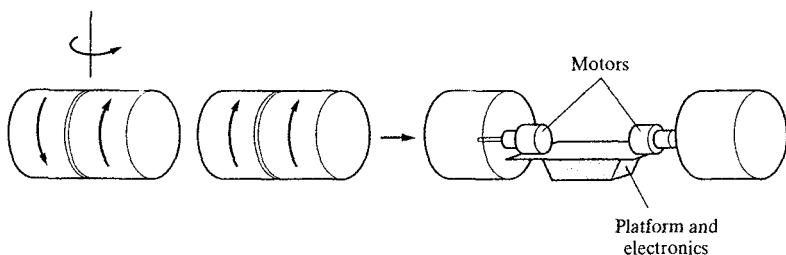


Figure 6.43 A schematic depiction of a possible arrangement for the cylindrical rover.

sembly, 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 will oscillate 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, will reduce the frequency of the oscillations. After you have built a prototype rover, you may add damping to the system, you may 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 the next chapter), and the added intelligence, can all be controlled by a microprocessor or other control circuitry. This design project is intentionally left open ended to allow you to be as creative as you wish. For 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.) Alternatively, you may use a microprocessor that responds to sensed information to do the same.

To complete the design project, first choose the cylinders, the platform size, the motors, the control system, and the sensors you want to incorporate in the design. You must make sure that 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 it 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.

6.13 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, these actuators can also be used in other devices that are used with robotics systems.

Although hydraulic systems are no longer common for industrial robots, they are still used for robots with a heavy payload and for devices that require 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 will 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. Parker Hanifin, *Step Motors and Servo Motors Control Catalog*, 1996–7.
3. McCormik, Malcolm, "A Primer on Brushless DC Motors," *Mechanical Engineering Magazine*, Feb. 1988, pp. 52–57.
4. Mazurkiewicz, John, "From Dead Stop to 2,000 rpm in One Millisecond," *Motion Control*, Sep./Oct. 1990, pp. 41–44.
5. Shaum, Loren, Richard C. Dorf, Editor, "Actuators," *International Encyclopedia of Robotics: Applications and Automation*, John Wiley and Sons, New York, 1988, pp. 12–18.
6. Martin, Fred, *Mini Board Technical Reference*, July 1992.
7. Oriental Motors U.S.A. Corporation, *Technical Information on Stepping Motors*.
8. Oriental Motor U.S.A. Corporation, *Application of Integrated Circuits to Stepping Motors*, 1987.
9. Shetty, Devdas, R. Kolk, *Mechatronics System Design*, PWS Publishing, Mass., 1997.
10. Auslander, David, C. J. Kempf, *Mechatronics: Mechanical System Interfacing*, Prentice Hall, Upper Saddle River, N.J., 1996.
11. Stiffler, Kent A., *Design with Microprocessors for Mechanical Engineers*, McGraw-Hill, New York, 1992.
12. J. Adolfsson, J. Karlsen, Editors, "Mechatronics '98, proceedings of the 6th UK Mechatronics Forum International Conference, Skovde, Sweden," *Pergamon Press*, Amsterdam, 1998.
13. Bolton, W., *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering*, 2nd Edition, Addison Wesley Longman, New York, 1999.
14. Ashley, S., "Magnetostrictive Actuators," *Mechanical Engineering*, June 1998, pp. 68–70.
15. "Push/Pull Magnetostrictive Linear Actuators," *NASA Tech Briefs*, August 1999, pp. 47–48.
16. "Tiny Steps for a Big Job," *NASA Motion Control Tech Briefs*, August 1999, pp. 1b–4b.
17. Toki America Technologies BioMetal Reference.
18. Erdman, Arhtur, G. N. Sandor, *Mechanism Design: Analysis and Synthesis*, Prentice-Hall, Englewood Cliffs, N.J., 1984.
19. "Hollow Shaft Actuators with Harmonic Drive Gearing," *NASA Motion Control Tech Briefs*, December 1998, pp. 10b–14b.
20. Orbidrive Catalog, Compudrive Corporation.

21. "Planetary Speed Reducer with Balls Instead of Gears," *NASA Tech Briefs*, October 1997, p. 15b.
22. Kedrowski, D., Scott Slimak, "Nutating Gear Drivetrain for a Cordless Screwdriver," *Mechanical Engineering*, January 1994, pp. 70-74.
23. "Travelling Wave Rotary Actuators: Piezoelectrically Generated Travelling Waves Drive Harmonic Gears," *NASA Tech Briefs*, October 1997, p. 10b.

PROBLEMS

1. A motor with rotor inertia of 0.030 Kgm^2 and maximum torque of 12 Nm is connected to a uniformly distributed arm with a concentrated mass at its end, as shown in Figure P6.1. 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) 5 and (b) 50.

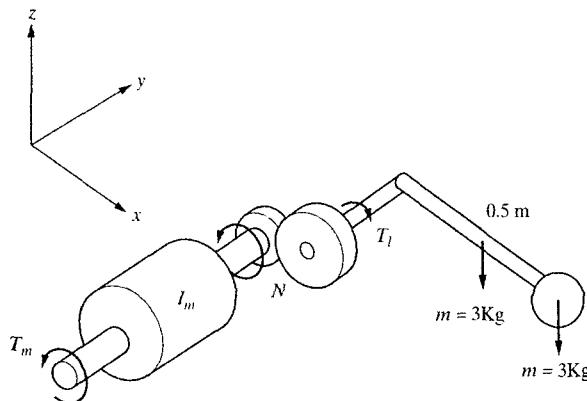


Figure P.6.1

2. Repeat Problem 1, but assume that the two gears have 0.002 Kgm^2 and 0.005 Kgm^2 inertias, respectively.
3. Estimate how much the torque-inertia ratio of a disk motor might be if it can go from zero to 2,000 rpm in one millisecond, and compare it to the motor of Problem 1.
4. Using a timer circuit, design a pulse generating circuit that will deliver a range of 5–500 impulses per second to a stepper motor driver.
5. Calculate the gear ratio for a Harmonic drive if $N_L = 100$, $N_F = 95$, $N_2 = 90$, $N_3 = 95$.
6. Write a program to generate a variable pulse stream to drive a motor with PWM voltages of 1, 2, 3, 4, and 5 volts for a 5-volt input.
7. Write a program to generate a sinusoidal PWM output for a constant input voltage.

7

Sensors

7.1 INTRODUCTION

In robotics, sensors are used for both internal feedback control and external interaction with the outside environment. Animals and humans have similar distinct sensors. For example, when you wake up, even before you open your eyes, you still can feel and 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 what the state of each muscle is. Similarly, in a robot, as the links and joints move, sensors such as potentiometers, encoders, and resolvers send signals to the controller that allow it to determine where each joint is. Additionally, as humans and animals possess senses of smell, touch, taste, hearing, vision, and speech to communicate with the outside world, robots may possess similar sensors that allow them to communicate with the environment. In certain cases, the sensors may be similar in function to that of the humans, such as vision, touch, and smell. In other cases, the sensors may be something humans do not have, such as a radioactive sensor.

There are many different types of sensors available that measure different phenomena. However, in this chapter, we will only discuss sensors that are used in conjunction with robotics and automatic manufacturing.

7.2 SENSOR CHARACTERISTICS

To choose an appropriate sensor for a particular need, one has to consider a number of different characteristics. These characteristics determine the performance, economy, ease of application, and applicability of the sensor. In certain situations,

different types of sensors may be available for the same purpose. In all these cases, the following may be considered 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, and life.
- **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. In addition, a large sensor may limit joint ranges. Thus, it is important to ensure that there is enough room for the joint sensors.
- **Weight:** Since robots are dynamic machines, the weight of the sensors is very important. A heavy sensor adds to the inertia of the arm, as well as reduces its overall payload.
- **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 will have to be converted to 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 circuits become necessary.
- **Resolution:** Resolution is the minimum step size within the range of measurement of the sensor. In a wire-wound potentiometer, it will be equal to the resistance of one turn of the wire. In a digital device with n bits, the resolution will be

$$\text{Resolution} = \frac{\text{Full Range}}{2^n}. \quad (7.1)$$

For example, an absolute encoder with 4 bits can report positions up to $2^4 = 16$ different levels. Thus, 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 the same change in output. Almost all devices in nature are somewhat nonlinear, with varying degrees of nonlinearity. Certain devices can be assumed to be linear within a cer-

tain range of their operation. Others may be linearized through assumptions. If an output is not linear, but its nonlinearity is known, the nonlinearity may be overcome by proper modeling, addition of equations, or additional electronics. For example, suppose that a displacement sensor has an output that is varying with the sine of an angle. Then to use the sensor, the designer may divide the output by the sine of the angle, either in programming or through addition of a simple electronic circuit that divides the signal by the sine of the angle. Thus, 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.
- **Frequency response:** Suppose that you attach a very-high-quality radio tuner to a small, cheap speaker. Although the speaker will reproduce the sound, its quality will be very low, whereas a high-quality speaker system with 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 small, cheap speaker. For example, since the natural frequency of a small speaker is higher, it can only reproduce higher frequency sounds. On the other hand, the speaker system with at least two speakers will run the signal into both tweeter and woofer speakers, one with high natural frequency and one with low natural frequency. 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 forcing frequency decreases or increases from the natural 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. Similarly, it is important to consider the frequency response of a sensor and determine whether the sensor's response is fast enough under all operating conditions.
- **Reliability:** Reliability is the ratio of how many times a system operates properly divided by how many times it is tried. For continuous, satisfactory operation, it is necessary to choose reliable sensors that last a long time, while considering the cost, as well as other requirements.
- **Accuracy:** Accuracy is defined as how close the output of the sensor is to the expected value. If for a given input, the output is expected to be a certain value, the accuracy is related to how close the sensor's output is to this value.
- **Repeatability:** If the sensor's output is measured a number of times in response to the same input, the output may be different each time. Repeatability is a measure of how varied the different outputs are relative to each other.

Generally, if enough tries are made, a range can be defined to include all results around the nominal value. This range is defined as repeatability. In general, repeatability is more important than accuracy, since in most cases, inaccuracies are systematic, and can be corrected or compensated, because they can be predicted and measured. Repeatability is generally random and cannot be easily compensated.

The following is a review of some sensors used in robotics, mechatronics, and automation.

7.3 POSITION SENSORS

Position sensors are used to measure displacements, both rotary 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:

7.3.1 Potentiometers

A potentiometer converts position information into a variable voltage through a resistor. As the sweeper on the resistor moves due to a change in position, the proportion of the resistance before or after the point of contact with the sweeper compared with the total resistance varies (Figure 7.1). Since in this capacity the potentiometer acts as a voltage divider, the output will be proportional to the resistance as

$$V_{\text{out}} = V_{\text{cc}} \frac{R_1}{R} \quad (7.2)$$

Potentiometers can be rotary or linear and thus can measure linear or rotary motions. Rotary potentiometers can also be multiple-turn, enabling the user to measure many revolutions of motion.

Potentiometers are either wire wound or thin film deposit (also called conductive plastic), which is a deposit of a thin film of resistive material on a surface. The major benefit of thin film potentiometers is that their output is continuous and thus 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 alone or together with

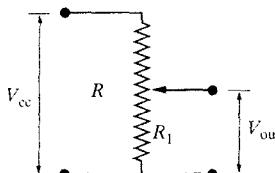


Figure 7.1 A potentiometer as a position sensor.

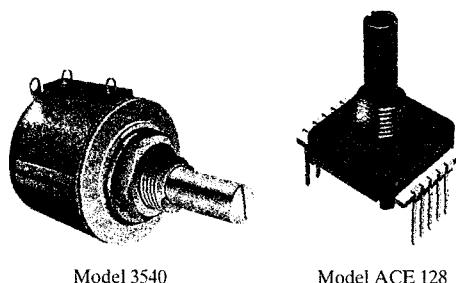


Figure 7.2 Typical commercial rotary potentiometers. Printed with permission from Bourns, Inc.

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 for a minimal input requirement, but maximum accuracy. This will be discussed in more detail in Section 7.3.2. Figure 7.2 shows two typical commercial potentiometers.

7.3.2 Encoders

An encoder is a simple device that can output a digital signal for each small portion of a movement. To do this, the encoder wheel or strip is divided into small sections, as shown in Figure 7.3. Each section is either opaque or clear. (It can also be either reflective or nonreflective.) A light source, such as an LED, on one side provides a beam of light to the other side of the encoder wheel or strip, where it is seen by another light-sensitive sensor, such as a phototransistor. If the wheel's angular position (or in the case of a strip, the linear position) is such that the light can go through, the sensor on the opposite side will be turned on and will have a high signal. If the angular position of the wheel is such that the light is stopped, the sensor will be off, and its output will be low (thus, a digital output). As the wheel rotates, it can continuously send signals. If the signals are counted, the approximate total angular displacement of the wheel can be measured at any time.

There are two basic types of encoders: Incremental and absolute. Figure 7.3 is an incremental encoder. In this type of encoder, the areas (arcs) of clear and

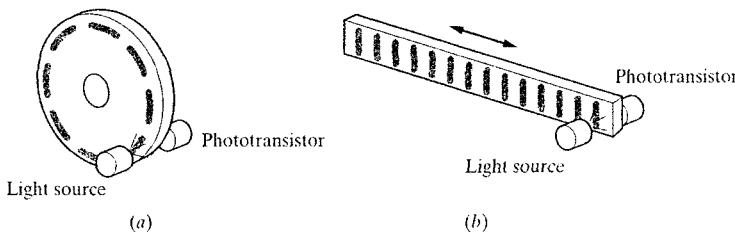


Figure 7.3 (a) A simple rotary incremental encoder wheel. This encoder measures angular rotations. (b) A linear incremental encoder, which can measure linear movements.

opaque sections are all equal and repeating. Since all arcs are the same size, each arc indicates a certain angle of revolution, which are all the same. If the wheel is divided into only two portions, each portion 180 degrees, its resolution will also be 180 degrees, and 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. Thus, the resolution of an optical encoder is related to the number of arcs of clear and opaque areas. Typical incremental encoders can have 512 to 1,024 arcs in them, reporting angular displacements with a resolution of 0.7 to 0.35 degree.

Optical encoders are either opaque wheels with removed material for clear areas (by drilling, cutting, and otherwise removing) or clear material such as glass with printed opaque areas. Many encoder wheels 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 wheel.

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 indicate directly what the actual value of the position is. 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. Thus, depending on where the starting point of a device, e.g. a robot, is, the final position may be different. 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 thus it is integrating the position signal. Unless the controller knows the startup 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. The controller will then know the displacements at all times, as long as the reset position is known.

Figure 7.4 shows the output of an incremental encoder. If only one set of arcs is used, it will be impossible to determine whether the wheel is rotating clockwise or counterclockwise. To remedy this, encoder wheels have two sets of arcs (two channels), 1/2 step out of phase with each other. As a result, the output signals of the two sets of arcs are also 1/2 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, the controller can be designed or programmed to determine the direction of rotation.

It is actually possible to double the resolution of the output of incremental encoders without increasing the number of arcs. To do this, instead of determining the change of signal only at the leading edges from low to high, we can also count the change at the trailing edges from high to low, which is equivalent to counting all changes. With two channels and double counting, a resolution four times as good can be achieved.

An alternative to incremental optical encoders is an absolute encoder. Each portion of the encoder wheel's angular displacement has a unique combination of clear and opaque arcs which give it a unique signature. Through this unique signature, it is possible to determine the exact position of the wheel at any time, without the need for a starting position. In other words, even at start time, the controller can

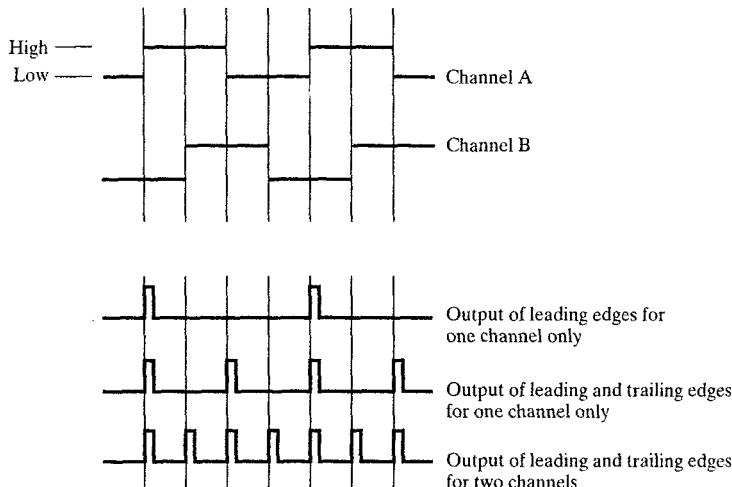


Figure 7.4 Output signals of an incremental encoder.

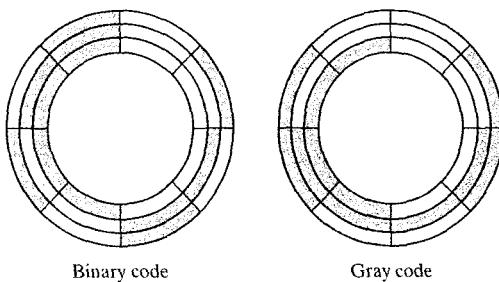


Figure 7.5 Each portion of the angular position of an absolute encoder has a unique signature. Through this signature, the angular position of the encoder can be determined.

determine the position of the wheel by considering the unique signature of the wheel at that location. As shown in Figure 7.5, there is a multiple row of arcs, each one different from the others. The first row may have only one clear and one opaque arc (one on, one off). The next row would have 4 (or 2^2), the next has 8 (or 2^3), etc. Each row must have its own light source and light sensor assembly. Each sensor assembly sends out one signal. Thus, two rows would require two inputs to the controller (2 bits), three rows would require 3 bits, etc.

As can be seen from the figure, an encoder with 4 sets of arcs 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. Thus, the resolution is only 22.5° . To increase the resolution, there would have to be more arcs, or bits. Although it is possible to have as many as 1,024 divisions (or even more on larger wheels) similar to an incremental encoder, the greater the number, the larger the number of bits is. So, for an encoder with up to 1,024 divisions on one arc,

TABLE 7.1 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

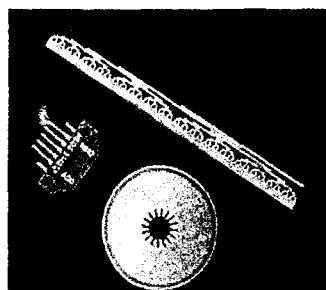
the controller would have to read $10 (1,024 = 2^{10})$ bits of information per encoder, which is huge. Just imagine that there are 6 encoders for the 6 joints, and thus, there may be 60 lines of information going to the controller for position only. Consequently, it is necessary to consider the advantages and disadvantages of incremental and absolute encoders.

Figure 7.5 also shows the difference between a binary code and a gray code. In binary code system, there are many instances where more than two of the bits change their 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, contrary to popular perception, the values of signals are not constantly read, but that the signal is measured (sampled) and held until the next sample reading. In binary code, where more than one signal bit changes at the same time, if all signals do not happen exactly at the same time, the sampling may not find all changes. In gray code, since there is only one change, the system will always find it. Table 7.1 lists the gray code for the first 12 numbers.

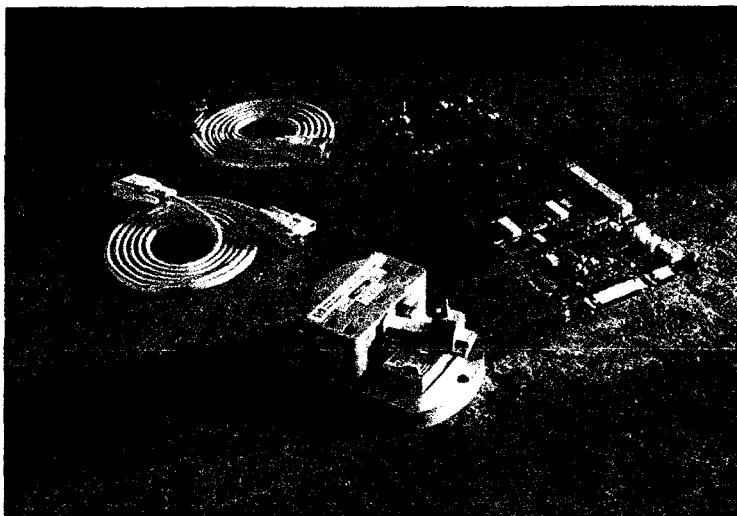
Optical encoders come in both linear and rotary versions, and except for their type of motion, they are exactly the same and work under the same principles. Figure 7.6 shows typical commercial encoders.

7.3.3 Linear Variable Differential Transformers (LVDT)

A linear variable differential transformer (or transducer) is actually a transformer whose core moves along with the distance being measured and that outputs a variable analog voltage as a result of this displacement. In general, a transformer is a device that converts an electrical energy into the same form of energy, but changes the voltage - current ratio. Except for losses, the total input energy to the device is the same as the total output energy. Since a transformer can increase or decrease voltage based on the number of turns in its coils, the corresponding current changes inversely with voltage. This occurs because there are two coils with different number of turns in their windings. The electrical energy into one coil creates a flux, which induces a voltage in the second coil proportional to the ratio of the number of turns in



(a)



(b)

Figure 7.6 (a) A typical commercial incremental encoder-wheel and absolute encoder-strip. (b) A typical encoder system. Reprinted with permission from Agilent Technologies, Inc.

the windings. The larger the number of turns in the secondary coil, the larger the voltage is, and, consequently, smaller the current is. However, the induction of voltage in the secondary is very much 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 will be minimal. In the presence of an iron core, the flux lines are gathered inward, increasing the strength of the field and thus the induced voltage. This is used to create the output variable voltage in the linear variable differential transformer (LVDT), as shown in Figure 7.7. The output of an LVDT is very linear, proportional to the input position of the core.

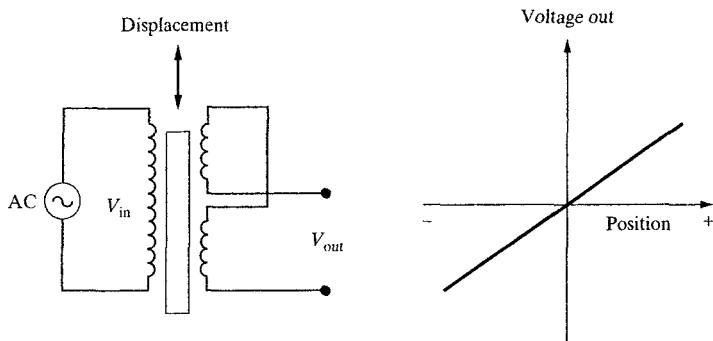


Figure 7.7 Linear variable differential transformer.

7.3.4 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 through slip rings (Figure 7.8). 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 other secondary coil that is perpendicular to it does not develop any voltage. 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.

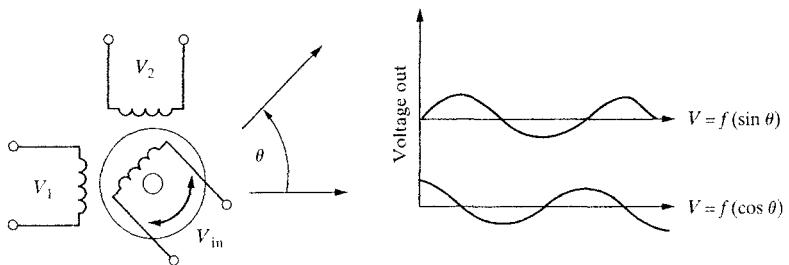


Figure 7.8 Schematic of a resolver.

7.3.5 Time-of-Travel (Magneto Reflective) Displacement Sensor

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 the distance if the speed of travel is known. By attaching the moving part to either the magnet or the conductor, the displacement can be measured. A simple schematic of the sensor is shown in Figure 7.9. The IBM 7565 robot displacement sensors are of this type, as shown in Figure 7.10.

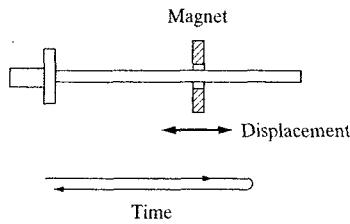


Figure 7.9 Schematic drawing of a time-of-travel displacement sensor

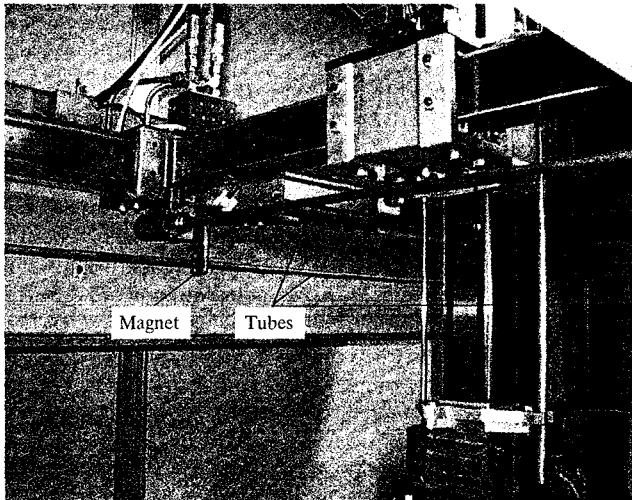


Figure 7.10 The IBM 7565 magneto reflective position sensor. A pulse generated at one end of a long tube travels within the tube until it reaches a magnet and bounces back. The time of travel of the signal is converted to position information.

7.4 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.

7.4.1 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), we can calculate velocity. A smaller length of time (dt) yields a more accurate calculated velocity, one that is closer to the true instantaneous velocity. However, if the rate of rotation of the encoder is low, the velocity measurement may become inaccurate. This velocity calculation is accomplished by programming the controller to convert number of signals in a given length of time into velocity information.

7.4.2 Tachometers

A tachometer is a generator that converts mechanical energy into electrical energy. Its output is an analog voltage proportional to the input angular speed. It may be used along with potentiometers to estimate velocities.

7.4.3 Differentiation of Position Signal

If the position signal is clean, it is actually possible, and even simple, to differentiate the position signal and to convert it to velocity signal. To do this, it is necessary that the signal be as continuous as possible in order to prevent creation of large impulses in velocity signal. Thus, it is recommended that thin film plastic resistors be used for position measurement, since a wire-wound potentiometer's output is piecewise and unfit for differentiation. However, differentiation of a signal is always noisy and should be done very carefully. Figure 7.11 shows a simple R-C circuit with an op-amp that can be used for differentiation, where the velocity signal is

$$V_{\text{out}} = -RC \frac{dV_{\text{in}}}{dt}. \quad (7.3)$$

Similarly, the velocity (or acceleration) signal can be integrated to yield position (or velocity) signals:

$$V_{\text{out}} = -\frac{1}{RC} \int V_{\text{in}} dt. \quad (7.4)$$

7.5 ACCELERATION SENSORS

Accelerometers are very common sensors for measuring accelerations. However, in general, accelerometers are not used with industrial robots, since no acceleration is generally measured in these robots. However, recently, acceleration measure-

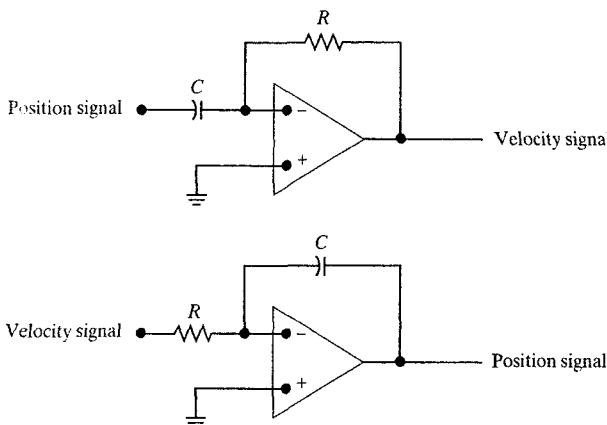


Figure 7.11 Schematics of differentiating and integrating R - C circuits with an op-amp.

ments have been used for high-precision control of linear actuators [2] and for joint-feedback control of robots [3].

7.6 FORCE AND PRESSURE SENSORS

7.6.1 Piezoelectric

Piezoelectric material compresses if exposed to a voltage and produces a voltage if compressed. This is used in 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.

7.6.2 Force-Sensing Resistor

The Force Sensing Resistor™ (FSR), manufactured by Interlink Electronics, is a polymer thick-film device that exhibits a decreasing resistance with increasing force applied normal to its surface. For a changing force of 10 to 10,000 gr, its resistance changes from about $500\text{ }\Omega$ to about $1\text{ K}\Omega$ [4]. Please also refer to reference [5] for UniForce™ sensors and [6] for others. Figure 7.12 shows a typical force-sensing resistor.

7.6.3 Strain Gauges

Strain gauges can also be used to measure forces. The output of the strain gauge is a variable resistance, proportional to the strain, which itself is a function of applied

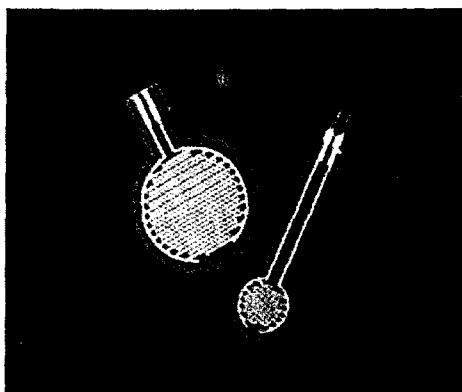


Figure 7.12 A typical force-sensing resistor. The resistance of this sensor decreases as the force acting on it increases.

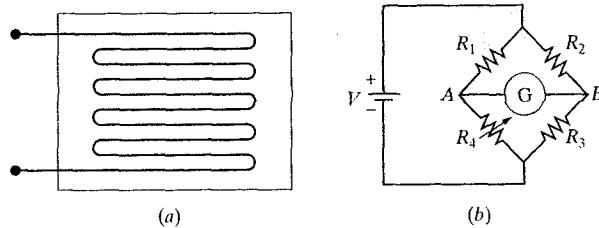


Figure 7.13 (a) A strain gauge and (b) a Wheatstone bridge.

forces. Thus, measuring the resistance, one can determine the applied force. Strain gauges are used to determine the forces at the end effector and the wrist of a robot. For example, the IBM 7565 robot has a set of strain gauges at the tip of its gripper, through which the forces applied by the gripper can be determined. A simple command allows the user to read the force and react accordingly. 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 7.13(a) is a simple schematic drawing of a strain gauge. Strain gauges are used within a Wheatstone bridge, as shown in Figure 7.13(b). A balanced Wheatstone bridge would have similar potentials at points *A* and *B*. If the resistance in any of the four resistors changes, there will be a current flow between these two junctions. Thus, it is necessary to first calibrate the bridge for zero flow in the galvanometer. Assuming that R_1 is the strain gauge, when under stress, its value will change, causing an imbalance in the Wheatstone bridge and a current flow between *A* and *B*. 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

$$\frac{R_1}{R_4} = \frac{R_2}{R_3}. \quad (7.5)$$

Strain gauges are sensitive to temperature changes. 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.

7.7 TORQUE SENSORS

Torque can be measured by a pair of strategically placed force sensors. Suppose that two force sensors are placed on a shaft, opposite of each other, on opposite sides. If a torque is applied to the shaft, it generates two opposing forces on the shaft's body, causing opposite direction strains. 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 7.14. Pure forces generate similar signals in a pair, while torques generate pairs of signals with opposite signs.

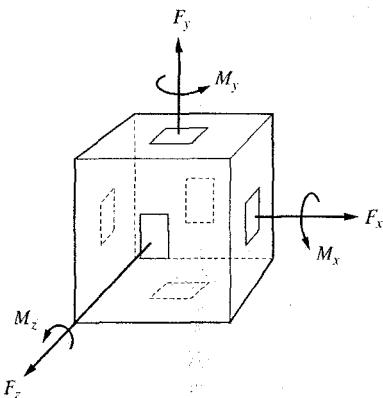


Figure 7.14 Arrangement of three pairs of strain gauges along the three major axes for force and torque measurements.

7.8 MICROSWITCHES

Microswitches, although extremely simple, are very useful and common in all robotics. They are used to cut off the electrical current through a conductor, and thus they 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.

7.9 LIGHT AND INFRARED SENSORS

These sensors react to the intensity of light projected on them by changing their electrical resistance. If the intensity of light is zero, the resistance is at maximum. The higher the intensity of light, the lower the resistance is and, consequently, the

higher the current is. As a result, the voltage drop across the switch is less. 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 minimal amount of light, it will turn on and otherwise will be 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 [7].

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 will not disturb humans if used in devices that project the light out. 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.

Simple infrared remote control devices are also available that can be used to establish remote-control communication links between devices and robots. Please refer to [6] for specifications.

7.10 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. For example, a strategically placed microswitch can send a signal to the controller if a mobile robot 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 how strong the touching force is.

A tactile sensor is a collection of touch sensors that 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 number of touch sensors are arranged in an array or matrix form, as shown in Figure 7.15. 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 the plunger moves in or out, it blocks the light from the LED projecting onto the light sensor. The output of the light sensor is then 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 LVDT's, pressure sensors, magnetic sensors, etc.

As the tactile sensor comes in contact with an object, depending on the shape and size of the object, different touch sensors react differently at a different se-

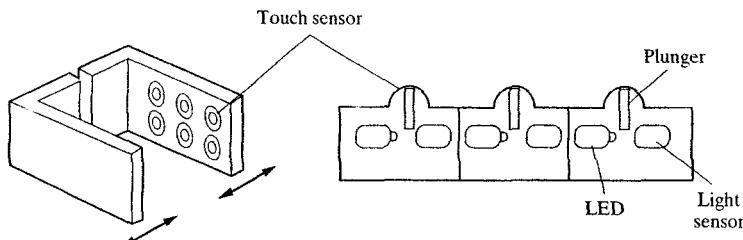


Figure 7.15 Tactile sensors are generally a collection of simple touch sensors arranged in an array or a matrix form with a specific order to relay contact and shape information to the controller.

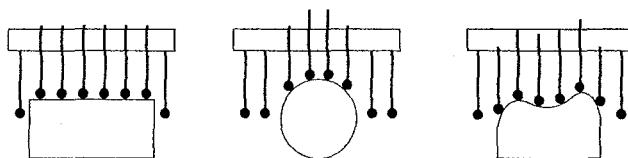


Figure 7.16 A tactile sensor can provide information about the object.

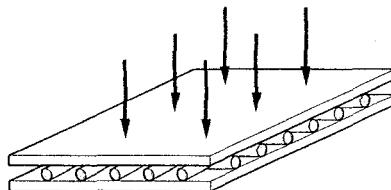


Figure 7.17 Skinlike tactile sensor.

quence. This information is then used by the controller to determine the size and the shape of the object. Figure 7.16 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 to detect it.

Attempts have also been made to create somewhat of a continuous skinlike tactile sensor that could function similar to human skin. In most cases, the design revolves around a matrix of sensors that are embedded between two polymer type layers, separated by an isolator mesh, as shown schematically in Figure 7.17. As a force is applied to the polymer, it is distributed between a few surrounding sensors, where each one sends a signal proportional to the force applied to it. For low resolution, these tactile sensors work satisfactorily [8].

Flex sensors can be used to determine flexing of a surface. This sensor sends a signal as it is flexed and can be used in virtual reality applications, in touch sensors, and other similar applications [6]. A similar sensor was used in a virtual-reality glove to measure the angles of finger joints [9]. The sensor is made of strips of urethane-based synthetic rubber filled with conductive carbon particles. The resistance of the

conductive elastomer decreases as it is stretched, thus providing a signal proportional to the angle joints.

7.11 PROXIMITY SENSORS

A proximity sensor is used to determine that an object is close to another object before contact is made. This noncontact 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:

7.11.1 Magnetic Proximity Sensors

These sensors are activated when they are close to a magnet. They can be used for measuring rotor speeds (and number of rotations), as well as for turning on or off a circuit [6]. Magnetic sensors may also be used to count number of rotations of wheels and motors, and thus they can be used as position sensors as well. 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 having the sensor stationary 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 send a signal when the door of the machine is open, and thus the controller stops the rotating or moving parts.

7.11.2 Optical Proximity Sensors

Optical proximity sensors consist of a light source called an emitter (either internal or external to the sensor) and a receiver, which senses the presence or the 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, the sensor is set up such that the light, emitted by the emitter, is not received by the receiver, unless an object is close-by. Figure 7.18 is a schematic drawing of an optical proximity sensor. Unless a reflective object is with-

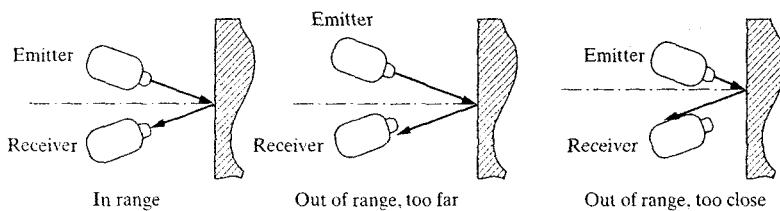


Figure 7.18 Optical proximity sensor.

in the range of the switch, the light is not seen by the receiver, and, therefore, there will be no signal.

7.11.3 Ultrasonic Proximity Sensors

In these sensors, an ultrasonic emitter emits frequent bursts of high-frequency sound waves (usually in the 200-kHz range). There are two modes of operations for ultrasonic sensors, namely, 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 will be sensed and a signal is produced. Otherwise, the receiver will 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 or the 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. Please refer to Section 7.12.1 for more information about ultrasonic sensors. Figure 7.19 is a schematic drawing of this type of sensor.

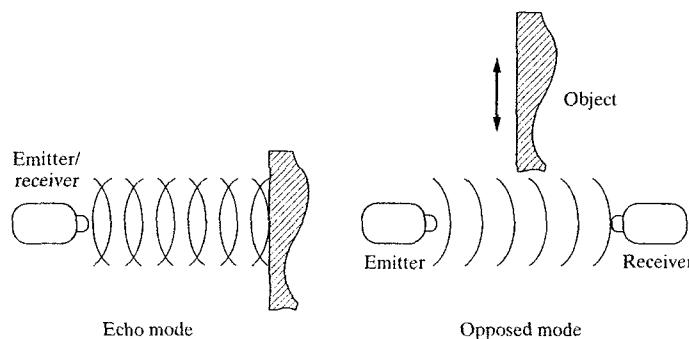


Figure 7.19 Ultrasonic proximity sensors.

7.11.4 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 detects the change and turns the solid state switch off. When the part leaves the range of the sensor, it turns on again.

7.11.5 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, the material acts as a capacitor, raising the total

TABLE 7.2 DIELECTRIC CONSTANTS FOR SOME SELECT MATERIALS

Air	1.000	Porcelain	4.4–7
Aqueous solutions	50–80	Cardboard	2–5
Epoxy resin	2.5–6	Rubber	2.5–3.5
Flour	1.5–1.7	Water	80
Glass	3.7–10	Wood, dry	2–7
Nylon	4–5	Wood, wet	10–30

capacitance of the sensor's probe. This will trigger an internal oscillator to turn on the output unit, which will send out an output signal. Thus, the sensor can detect the presence of an object within a range. Capacitive sensors can detect nonmetal materials such as wood, liquids, and chemicals. Table 7.2 shows dielectric constants for some select materials.

7.11.6 Eddy Current Proximity Sensors

As was mentioned in Chapter 5, when a conductor is placed within a changing magnetic field, an electromotive force (emf) is induced in it, which 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 of the first coil's flux, 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 material, as well as nondestructive testing of voids and cracks, thickness of materials, etc.

7.12 RANGE FINDERS

Unlike proximity sensors, range finders are used to find larger distances, to detect obstacles, and to map 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 ray of light that forms a spot on the object. The spot is seen by a receiver such as a camera or phototransistor. The range or depth is calculated from the triangle formed between the receiver, the light source, and the spot on the object, as in Figure 7.20.

As is evident from Figure 7.20(a), the particular arrangement between the object, the light source, and the receiver only happens at one instant. At this point, the distance d can be calculated by

$$\tan \beta = \frac{d}{l_1}, \quad \tan \alpha = \frac{d}{l_2}, \quad \text{and} \quad L = l_1 + l_2.$$

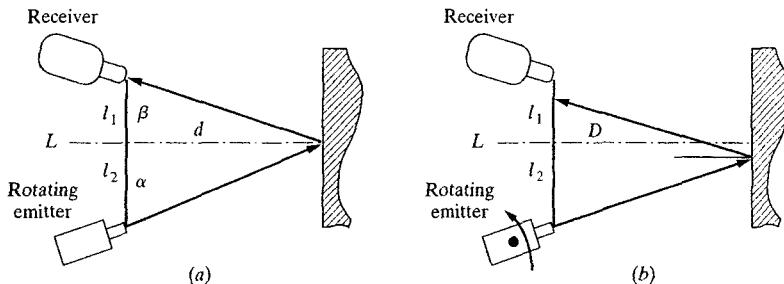


Figure 7.20 Triangulation method for range measurement. The receiver will only detect the spot on the object when the emitter is at a particular angle, which is used to calculate the range.

Substituting and manipulating the equation yields

$$d = \frac{L \tan \alpha \tan \beta}{\tan \alpha + \tan \beta} \quad (7.6)$$

Since L and β are known, if α is measured, d can be calculated. You can see from Figure 7.20(b) that except at that instant, the receiver will not see the reflected light. As a result, it is necessary to rotate the emitter, and as soon as the reflected light is observed, 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.

7.12.1 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, and in robots for navigation and range measurement. Their disadvantage is in their limited resolution, which is limited by the wavelength of the sound and natural inhomogeneities 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. Current 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 this technique, a transducer emits a pulse of high-frequency ultrasound, which travels a certain distance and is reflected back when it encounters a separation in the medium; it is then received by the receiver. The distance between the transducer and the object is half the distance traveled, which is equal to the time-of-

flight times the speed of sound. Of course, the accuracy of the measurement not only depends on the wavelength of the signal, but also to 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), the density of the medium, and the 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 the 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. Thus, higher frequency ultrasound devices yield a 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. It should be mentioned that 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. Thus, there is a trade-off between the lateral resolution and signal attenuation in relation with the beam frequency.

Background noise is another problem with ultrasonics. 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. Thus, it has been recommended to use frequencies above 100 kHz in industrial environments.

Ultrasonics can be used for distance measurement, mapping, and flaw detection. A single-point distance measurement is called spot checking, as opposed to range array acquisition for multiple-data-point-acquisition techniques used for three-dimensional mapping. In this case, a large number of distances to different locations on an object are measured. The collection of distance data provides a three-dimensional map of the surface of the object. It should be noted that since only half the surface area of a three-dimensional object can be ranged, these measurements are also referred to as two-and-one-half-dimensional. The backside of the object or areas obscured by other parts cannot be ranged.

7.12.2 Light-Based Range Finders

Light-based (including laser-based) 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 should not divert) to travel to an object and back, much similar to an ultrasonic sensor. Since the speed of light in air is 186,000 miles/s, it travels 1 ft in 1 ns. To have a reso-

lution of 0.25 inch, the instrumentation should have a resolution of 21 ps. This is done using the time spectroscopy technique used in nuclear science and high-energy physics.

In one indirect method, the time delay is measured by modulating a long burst of light with a low-frequency sinusoidal wave using a time-to-amplitude converter (TAC) and measuring the phase difference between the modulations between the emitted light and the backscattered light. This, in effect, slows down the wave speed to measurable scales by substituting the speed of light with low speed modulations, but still takes 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 will be discussed later in Chapter 8. A variation to this technique involves the use of a small laser diode along with a single camera [10]. 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.

7.13 SNIFF SENSORS

Sniff sensors are similar to smoke detectors. They are sensitive to particular gases and send a signal when they detect those gases. They are used for safety purposes as well as for search and detection purposes [11,12].

7.14 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 8. However, you must realize that vision systems are, in fact, sensors and that they relate the function of a robot to its environment as all other sensors do.

7.15 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 the 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 will reconstruct the original signal if combined. We will discuss this in more detail in Chapter 8. However, it is useful to realize that all signals will have major frequencies that constitute a particular spectrum and that this spectrum is generally different for other signals. In voice-recognition systems, it

is assumed that every word (or letter or sentence), when decomposed into its constituent frequencies, will have a unique signature composed of its major frequencies, which allow the system to recognize the word.

To do this, the user must train the system by speaking the words a priori to allow the system to create a lookup table of the major frequencies of the spoken words. Later, when a word is spoken and its frequencies determined, the result is compared with the lookup 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, the more accurate the frequencies, the narrower the allowable variations. This means that if the system tries to match many frequencies for better accuracy, in the presence of any noise or any variations in the spoken words, the system will not be able to recognize the word. On the other hand, if a limited number of frequencies is matched in order to allow for variations, then it may mix the words with other similar words. A universal system that recognizes all accents and variations in speaking may not be either possible or useful. Many robots have been equipped with voice-recognition systems in order to communicate with the users. In most cases, the robot is trained by the user and it 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 will send a signal to the controller, which, in turn, will run the robot to the desired location and orientation. This has been particularly useful with robots that are used to aid the disabled, as well as for medical robots.

7.16 VOICE SYNTHESIZERS

Voice synthesis is accomplished in two different ways. One 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 chip and a corresponding program. Although this type of system can reproduce any word, it sounds unnatural and machine-like. For example of the difficulty encountered by this kind of system, consider the two words “power” and “mower.” As you see, although these two words are written very similarly, they are pronounced very differently. This kind of a system will not be able to recognize this.

The alternative is to record the words that the system may need to synthesize and to access them from memory or tape as needed. Telephone time 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 that the machine needs to say are known a priori, this system can be used.

With advances in computer technology, voice recognition and synthesis will be advanced significantly in the future.

7.17 REMOTE CENTER COMPLIANCE (RCC) DEVICE

Although remote center compliance (RCC) devices are not actual sensors they are discussed here, because they act as sensing devices for misalignments and provide a

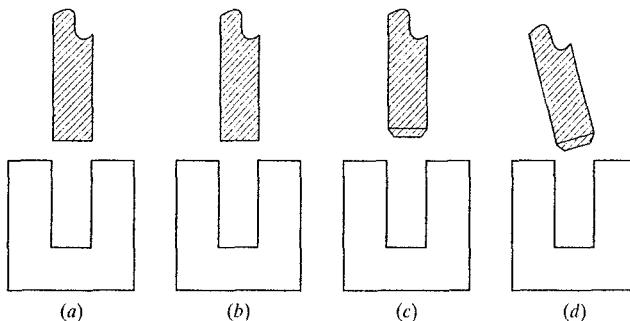


Figure 7.21 Misalignment of assembling elements.

means of correction for robots. However, RCC devices are completely passive, and there are no input or output signals.

An RCC device is an attachment that is added to the wrist of 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 of the robot.

Suppose that a robot is to push a peg into a hole in a part, as shown in Figure 7.21. If the hole and the peg are exactly the right sizes and if they are exactly aligned — both laterally and axially — then the robot will be able to push the peg into the hole. However, in most cases, this is impossible to achieve. Imagine that the hole is slightly off such that the centerline of the hole and the peg are a small distance apart, as in Figure 7.21(b).

If the robot is in position-control mode, it will 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. The stiffer the robot, a sign of a good robot, the worse this problem becomes. If the robot has some compliance, it is actually possible to cut a chamfer (Figure 7.21 (c)) 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. Alternatively, 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 7.21(d)). In this case, even if the peg and the hole are exactly aligned at the entrance of the hole, if the peg is pushed in, one of the two will have to either deflect or break. Once again, if either the part or the robot is allowed to move (a compliant robot), the problem can be avoided. However, in either of the two cases, a compliant robot that will give way enough to prevent breakage will probably have unacceptable accuracy.

What is needed is to have a device that can add selected compliance to the robot end effector, so that it will allow the robot to have selective compliance in the directions needed, without affecting its accuracy in other directions. An RCC device provides this selective compliance through a simple four-bar mechanism.

To understand how the RCC device works, consider the simple four-bar mechanism shown in Figure 7.22. In any mechanism, there are a total of $M = n \times (n - 1)/2$ instantaneous centers of zero velocity, where n is the number of links,

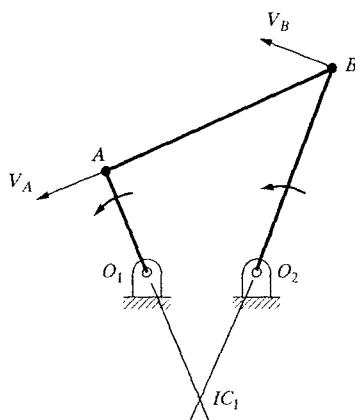


Figure 7.22 Instantaneous centers of zero velocity for a four-bar mechanism.

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 four-bar mechanism, there will be 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\$.

To find the instantaneous center of zero velocity for the coupler (which we are interested in 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 $\vec{V} = \bar{\omega} \times \vec{r}$, the velocity of any point is normal to its radius of curvature \$\vec{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 a zero instantaneous velocity, it means that at this instant, it is not moving, and thus 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 thus its acceleration cannot be zero.

Now consider the parallelogram four-bar mechanism shown in Figure 7.23(a). Since the two normals to the two velocities at \$A\$ and \$B\$ are parallel, the instantaneous center of zero velocity for the coupler will be at infinity, indicating that the coupler is not rotating, but is in pure translation. This means that the coupler will always translate to the left or right without any rotation (although its motion is curvilinear). Figure 7.23(b) shows a four-bar mechanism with two links of equal length and the instantaneous center of zero velocity for its coupler, which allows an instantaneous rotation of the coupler link about the \$IC\$. As you can imagine, these two mechanisms can provide simple translation or rotation about a remote center when needed. An

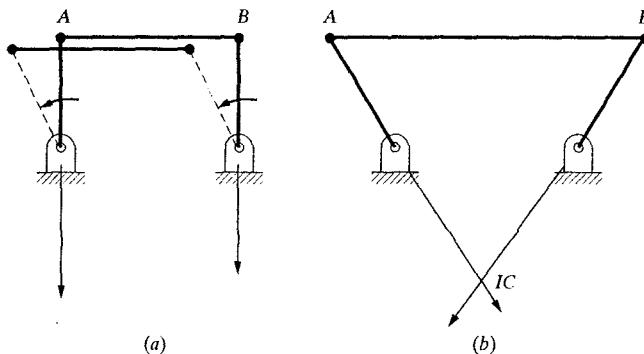


Figure 7.23 Special four-bar mechanisms, the basis for an RCC device.

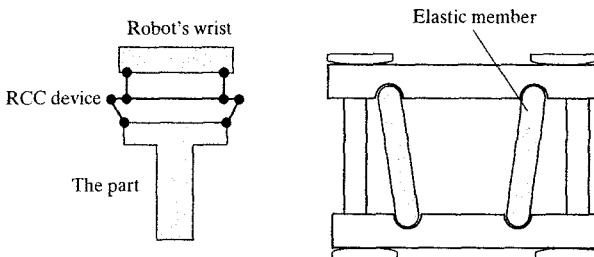


Figure 7.24 Schematic of an RCC device.

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 (thus 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, you should realize that this compliance is only lateral (or angular), where it is needed. However, 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 robot's stiffness and, thus, its accuracy.

Figure 7.24 is a schematic drawing of a RCC device. It consists of two rigid plates, where shear columns provide axial stiffness while providing lateral compliance. 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 center location relative to the center of the device. Thus, there may be need for multiple RCC devices if more than one part or operation is performed, and must be picked accordingly [13].

7.18 DESIGN PROJECT

At this point, you may want to incorporate as many sensors as you desire, or have available to you, in your robot. Some of the sensors will be necessary for feedback,

which are essential if you are to control the robot. Others are added based on need and availability. This is a very interesting part of any robotics project. You may experiment with different sensors for different applications and even come up with your own sensors. For example, the antistatic foam that is used for transporting IC chips and other electronic circuits is actually conductive and sensitive to pressure. As a result, it may be used to make very simple, yet interesting, touch sensors for your robot. Just connect two wires to two ends of a piece of this foam and attach to a battery. As you compress the foam, its resistance and, as a result, its voltage drop will change. Measuring the current in the circuit will provide tactile information. You may experiment with other sensors that have not been mentioned here, but are available from electronic warehouses.

Similarly, you may integrate sensors to the rolling-cylinders rover for control and added intelligence. For example, visible light and infrared sensors located in the center of the platform will allow you to communicate with the rover by projecting a visible or infrared light beam through the gap between the cylinders. Proximity sensors and range finders can also be used to determine proximity or distance to walls and other obstacles and for navigating in different environments.

7.19 SUMMARY

In this chapter, we discussed a variety of different sensors that are used in conjunction with robots and robotics 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 and some are difficult to use, are expensive, and require much support circuitry. Each sensor has its own advantages and disadvantages. For example, an incremental *encoder* can provide simple, digital, position and velocity information with minimal input requirements. However, the absolute position cannot be measured with it. An absolute encoder provides absolute position information in digital form, but requires many lines of input 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 thus 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 wakeup and one to accurately report the changes in position. Together they provide all the information that is 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. "Higher Resolution Optoelectronic Shaft-Angle Encoder." *NASA Tech Briefs*, March 2000, pp. 46–48.
2. 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.

3. 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.
4. Interlink Electronics, Santa Barbara, California.
5. Force Imaging Technologies, Chicago, IL.
6. Jameco Electronics Catalog, Belmont California.
7. Puopolo, Michael G., Saeed B. Niku, "Robot Arm Positional Deflection Control with a Laser Light," Proceedings of the Mechatronics '98 Conference, Skovde, Sweden, Adolfsen and Karlsen, Editors, Pergamon Press, Sep. 98, pp. 281–286.
8. Hillis, Daniel, "A High Resolution Imaging Touch Sensor," *Robotics Research*, 1:2, MIT press, Cambridge, Mass.
9. "Glove Senses Angles of Finger Joints," *NASA Tech Briefs*, April 1998.
10. 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.
11. "Sensors that Sniff," High Technology, February 1985, p. 74.
12. "Electronic Noses Made From Conductive Polymer Films," *NASA Tech Briefs*, July 1997, pp. 60–61.
13. Lord Corporation Catalogs for Remote Center Compliance Devices.

Image Processing and Analysis with Vision Systems

8.1 INTRODUCTION

There is a very large body of work associated with vision systems, image processing, and pattern recognition that addresses many different hardware- and software-related topics. This information has been accumulated since the 1950s, and with the added interest in the subject from different sectors of the industry and economy, it is growing rapidly. The enormous number of papers published every year indicates that there must be many useful techniques constantly appearing in the literature. At the same time, it also means that a lot of these techniques may be unsuitable for other applications. In this chapter, we will study and discuss some fundamental techniques for image processing and image analysis, with a few examples of routines developed for certain purposes. The chapter does not profess to be a complete survey of all possible vision routines, but only an introduction. It is recommended that the interested reader continue studying the subject through other references.

The next few sections present some fundamental definitions of terms and basic concepts that we will use throughout the chapter.

8.2 IMAGE PROCESSING VERSUS IMAGE ANALYSIS

Image processing relates to the preparation of an image for later analysis and use. Images captured by a camera or a similar technique (e.g., by a scanner) are not necessarily in a form that can be used by image analysis routines. Some may need improvement to reduce noise, others may need to be simplified, and still others may need to be enhanced, altered, segmented, filtered, etc. *Image processing* is the collection of routines and techniques that improve, simplify, enhance, or otherwise alter an image.

Image analysis is the collection of processes in which a captured image that is prepared by image processing is analyzed in order to extract information about the image and to identify objects or facts about the object or its environment.

8.3 TWO- AND THREE-DIMENSIONAL IMAGES

Although all real scenes are three dimensional, images can either be two or three dimensional. Two-dimensional images are used when the depth of the scene or its features need not be determined. As an example, consider defining the surrounding contour or the silhouette of an object. In that case, it will not be necessary to determine the depth of any point on the object. Another example is the use of a vision system for inspection of an integrated circuit board. Here, too, there is no need to know the depth relationship between different parts, and since all parts are fixed to a flat plane, no information about the surface is necessary. Thus, a two-dimensional image analysis and inspection will suffice.

Three-dimensional image processing deals with operations that require motion detection, depth measurement, remote sensing, relative positioning, and navigation. CAD/CAM-related operations also require three-dimensional image processing, as do many inspection and object recognition tasks. Other techniques, such as computed tomography (CT) scan, are also three dimensional. In computed tomography, either X-rays or ultrasonics pulses are used to get images of one slice of the object at a time, and later, all of the images are put together to create a three-dimensional image of the internal characteristics of the object.

All three-dimensional vision systems share the problem of coping with many-to-one mappings 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 contrary, when the environment is unknown or noisy and uncontrolled (e.g., in underwater operations), the systems are not very accurate and require additional processing of the information. Thus, they operate at low speeds. In addition, a three-dimensional coordinate system has to be dealt with.

8.4 WHAT IS AN IMAGE?

An image is a representation of a real scene, either in black and white or in color, and either in print form or in a digital form. Printed images may have been reproduced either by multiple colors and gray scales (as in color print or halftone print) or by a single ink source. For example, in order to reproduce a photograph with real halftones, one has to use multiple gray inks, which, when combined, produce an image that is somewhat realistic. However, in most print applications, only one color of ink is available (such as black ink on white paper in a newspaper or copier). In that 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

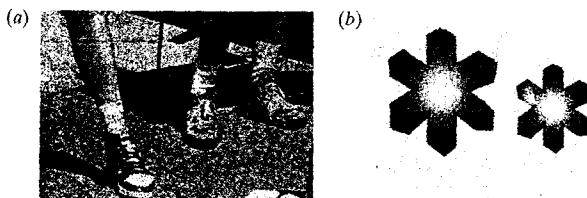


Figure 8.1 Examples of gray intensity creation in printed images. In print, only one color of ink is used, while the ratio of the black to the white area of the pixel is changed to create different gray levels.

into small sections. In each section, if the ink portion of the section is smaller compared to the white, blank area, the section will look lighter gray. (See examples in Figure 8.1.) If the black ink area is larger compared to the white area, it will look darker gray. By changing the size of the printed dot, many gray levels may be produced, and collectively, a gray-scale picture may be printed.

Unlike printed images, television and digital images are divided into small sections called *picture cells*, or *pixels* (in three-dimensional images, they are called *volume cells* or *voxels*), where the size of all pixels are the same, while the intensity of light in each pixel is varied to create the gray images. Since we deal with digital images, we will always refer to pixels of the same size with varying intensities.

8.5 ACQUISITION OF IMAGES

There are two types of vision cameras: analog and digital. Analog cameras are not very common any more, but are still around; they used to be standard at television stations. Digital cameras are much more common and are mostly similar to each other. A video camera is a digital camera with an added videotape recording section. Otherwise, the mechanism of image acquisition is the same as in other cameras that do not record an image. Whether the captured image is analog or digital, in vision systems the image is eventually digitized. In a digital form, all data are binary and are stored in a computer file or memory chip.

The following short discussion is about analog and digital cameras and how their images are captured. Although analog cameras are not common anymore, since the television sets available today are still mostly analog, understanding the way the camera works will help in understanding how the television set works. Thus, both analog and digital cameras are examined here.

8.5.1 Vidicon Camera

A vidicon camera is an analog camera that transforms an image into an analog electrical signal. The signal, a variable voltage (or current) versus time, can be stored, digitized, broadcast, or reconstructed into an image. Figure 8.2 shows a simple schematic of a vidicon camera. With the use of a lens, the scene is projected onto a screen made up of two layers: a transparent metallic film and a photoconductive mosaic that is sensitive to light. The mosaic reacts to the varying intensity of light by varying its resistance. As a result, as the image is projected onto it, the magnitude of the resistance at each location varies with the intensity of the light. An electron gun generates and sends a continuous cathode beam (a stream of electrons with a nega-

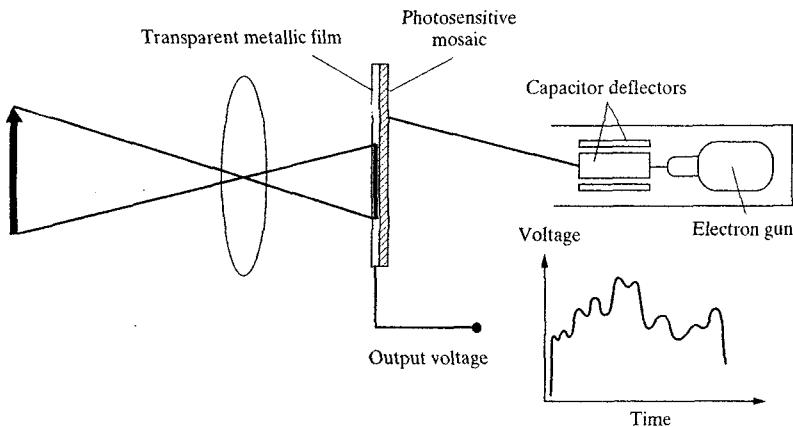


Figure 8.2 Schematic of a vidicon camera.

tive charge) through two pairs of capacitors (deflectors) that are perpendicular to each other. Depending on the charge on each pair of capacitors, the electron beam is deflected up or down, and left or right, and is projected onto the photoconductive mosaic. At each instant, as the beam of electrons hits the mosaic, the charge is conducted to the metallic film and can be measured at the output port. The voltage measured at the output is $V = IR$, where I is the current (of the beam of electrons), and R is the resistance of the mosaic at the point of interest.

Now suppose that we routinely change the charges in the two capacitors and thus deflect the beam both sideways and up and down, so as to cause it to scan the mosaic (a process called a *raster scan*). As the beam scans the image, at each instant the output is proportional to the resistance of the mosaic or proportional to the intensity of the light on the mosaic. By reading the output voltage continuously, an analog representation of the image can be obtained.

To create moving images in televisions, the image is scanned and reconstructed 30 times a second. Since human eyes possess a temporary hysteresis effect of about 1/10 second, images changing at 30 times a second are perceived as continuous and thus moving. The image is divided into two 240-line subimages, interlaced onto each other. Thus, a television image is composed of 480 image lines, changing 30 times a second. In order to return the beam to the top of the mosaic, another 45 lines are used, creating a total of 525 lines. In most other countries, 625 lines are the standard. Figure 8.3 depicts a raster scan in a vidicon camera.

If the signal is to be broadcast, it is usually frequency modulated (FM); that is, the frequency of the carrier signal is a function of the amplitude of the signal. The signal is broadcast and is received by a receiver, where it is demodulated back to the original signal, creating a variable voltage with respect to time. To re-create the image — for example, in a television set — this voltage must be converted back to an image. To do this, the voltage is fed into a cathode-ray tube (CRT) with an electron gun and similar deflecting capacitors, as in a vidicon camera. The intensity of the electron beam in the television is now proportional to the voltage of the signal, and is scanned similar to the way a camera does. In the television set, however, the beam

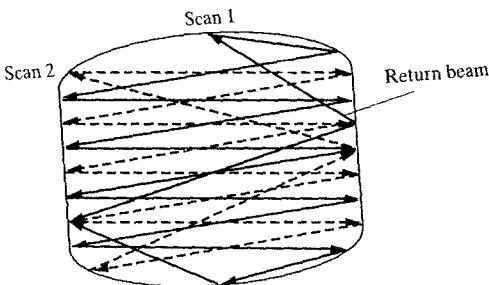


Figure 8.3 A raster scan depiction of a vidicon camera.

is projected onto a phosphorous-based material on the screen, which glows proportionally to the intensity of the beam, thus re-creating the image.

For color images, the projected image is decomposed into the three colors of red, green, and blue (RGB). The exact same process is repeated for the three images, and three simultaneous signals are produced and broadcast. In the television set, three electron guns regenerate three simultaneous images in RGB on the screen, except that the screen has three sets of small dots (pixels) that react by glowing in RGB colors and are repeated over the entire screen. All color images in any system are divided into RGB images and are dealt with as three separate images.

If the signal is not to be broadcast, it either is recorded for later use, is digitized (as discussed later), or is fed into a monitor for direct viewing.

8.5.2 Digital Camera

A digital camera is based on solid-state technology. As with 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 that has hundreds of thousands of extremely small photosensitive areas called *photosites* printed on it. Each small area of the wafer is a pixel. As the image is projected onto the image area, at each pixel location of the wafer a charge is developed that is proportional to the intensity of light at that location. (Thus, a digital camera is also called a charge coupled device, or CCD camera, and a charge integrated device, or CID camera). The collection of charges, if read sequentially, would be a representation of the image pixels. (See Figure 8.4).

The wafer may have as many as 520,000 pixels in an area with dimensions of a fraction of an inch ($\frac{3}{16} \times \frac{1}{4}$). Obviously, it is impossible to have direct wire connections to all of these pixels to measure the charge in each one. To read such an enormous number of pixels, 30 times a second the charges are moved to optically isolated shift registers next to each photosite, are moved down to an output line, and then are read [1,2]. The result is that every thirtieth of a second the charges in all pixel 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 8.5(a). Figure 8.5(b) is the CCD element of a VHS camera.

Similar to CCD cameras for visible lights, long-wavelength infrared cameras yield a televisionlike image of the infrared emissions of a scene [3].

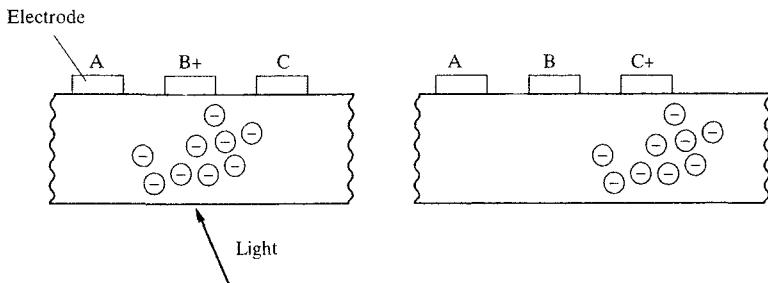


Figure 8.4 Image acquisition with a digital camera involves the development, at each pixel location, of a charge proportional to the light at the pixel. The image is then read by moving the charges to optically isolated shift registers and reading them at a known rate.

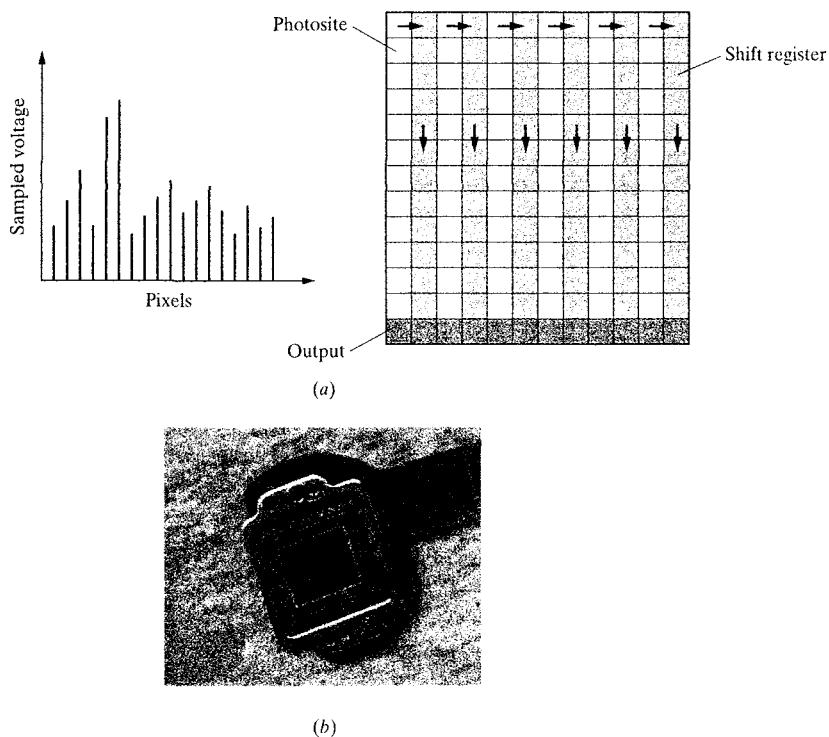


Figure 8.5 (a) Image data collection model. (b) The CCD element of a VHS camera.

8.6 DIGITAL IMAGES

The sampled voltages from the aforementioned process are first digitized through an analog-to-digital converter (ADC) and then either stored in the computer storage unit in an image format such as TIFF, JPG, Bitmap, etc., or displayed on a monitor. Since it is digitized, the stored information is a collection of 0's and 1's that represent the intensity of light at each pixel; a digitized image is nothing more than a computer file that contains the collection of these 0's and 1's, sequentially stored to represent the intensity of light at each pixel. The files can be accessed and read by a program, can be duplicated and manipulated, or can be rewritten in a different form. Vision routines generally access this information, perform some function on the data, and either display the result or store the manipulated result in a new file.

An image that has different gray levels at each pixel location is called a *gray image*. The gray values are digitized by a digitizer, yielding strings of 0's and 1's that are subsequently displayed or stored. A color image is obtained by superimposing three images of red, green, and blue hues, each with a varying intensity and each equivalent to a gray image (but in a colored state). Thus, when the image is digitized, it will similarly have strings of 0's and 1's for each hue. A binary image is an image such that each pixel is either fully light or fully dark — a 0 or a 1. To achieve a binary image, in most cases a gray image is converted by using the histogram of the image and a cut-off value called a *threshold*. A histogram determines the distribution of the different gray levels. One can pick a value that best determines a cutoff level with least distortion and use that value as a threshold to assign 0's (or "off") to all pixels whose gray levels are below the threshold value and to assign 1's (or "on") to all pixels whose gray values are above the threshold. Changing the threshold will change the binary image. The advantage of a binary image is that it requires far less memory and can be processed much faster than gray or colored images.

8.7 FREQUENCY DOMAIN VS. SPATIAL DOMAIN

Many processes that are used in image processing and analysis are based on 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 pixels of the image. As a result, each pixel is affected directly by the process. However, the two techniques are equally important and powerful and are used for different purposes. Note that although spatial- and frequency-domain techniques are used differently, they are related. For example, suppose that a spatial filter is used to reduce noise in an image. As a result, noise level in the image will be reduced, but at the same time, the frequency spectrum of the image will also be affected, due to the reduction in noise.

The next several sections discuss some fundamental issues about frequency and spatial domains. The discussion, although general, will help us throughout the entire chapter.

8.8 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 number of sines and cosines of different amplitudes and frequencies as follows:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos n\omega t + \sum_{n=1}^{\infty} b_n \sin n\omega t. \quad (8.1)$$

If you add these sines and cosines together again, you will have reconstructed the original signal. Equation (8.1) is called a *Fourier series*, and the collection of different frequencies present in the equation is called the *frequency spectrum* or *frequency content* of the signal. Of course, although the signal is in the amplitude–time domain, the frequency spectrum is in the amplitude–frequency domain. To understand this concept 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 of only one frequency with a constant amplitude, if we were to plot the signal in the frequency domain, it would be represented by a single line at the given frequency, as shown in Figure 8.6. Obviously, if we plot the function represented by the arrow in Figure 8.6(b) with the given frequency and amplitude, we will have reconstructed the same sine function. The plots in Figure 8.7 are similar and represent $f(t) = \sum_{n=1,3,\dots,15}(1/n) \sin(nt)$. The frequencies are also plotted in the frequency–amplitude domain. Clearly, when the number of frequencies contained in $f(t)$ increases, the summation becomes closer to a square function.

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 an impulse, a pulse, a square wave, or anything else similar to them or modeled by them) have a large number of frequencies. The sharper the change, the higher is the number of frequencies needed to reproduce it. Thus, any video (or other) signal that contains sharp changes (such as noise, high contrasts, or an impulse or step function) or that has

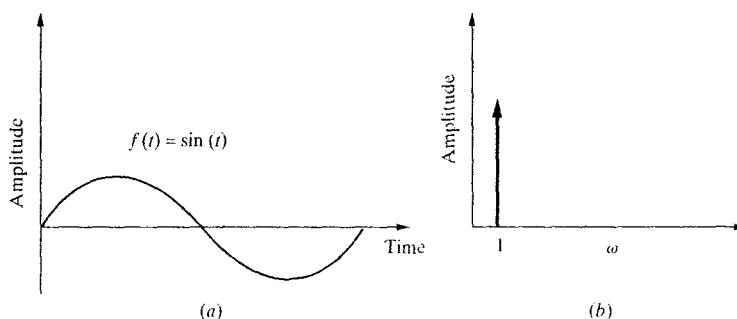


Figure 8.6 Time-domain and frequency-domain plots of a simple sine function.

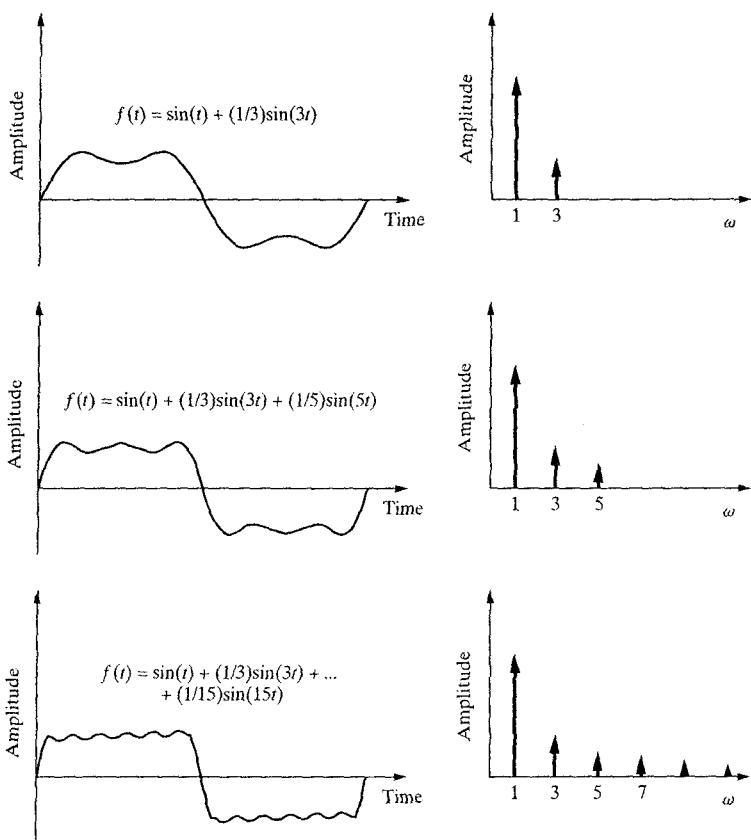


Figure 8.7 Sine functions in the time and frequency domain for a successive set of frequencies. As the number of frequencies increases, the resulting signal becomes closer to a square function.

detailed information (high-resolution signals with fast, varying changes) will have a larger number of frequencies in its frequency spectrum.

A similar analysis can be applied to nonrepeating signals as well. (The equation used is a *Fourier transform* or, sometimes, a *fast Fourier Transform*, or *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 will be infinite frequencies in the spectrum, generally, some of the major frequencies within the spectrum will have larger amplitudes. These major frequencies, or *harmonics*, are used in identifying and labeling a signal, including recognizing voices, shapes, objects, etc.

8.9 FREQUENCY CONTENT OF AN IMAGE: NOISE, EDGES

Consider sequentially plotting the gray values of the pixels of an image (on the y -axis) against time or pixel location (on the x -axis) as the image is scanned. (See Section 8.5.) The result will be a discrete time plot of varying amplitudes showing the intensity of light at each pixel, as indicated in Figure 8.8. Let's say that we are on the ninth row and are looking at pixels 129–144. The intensity of pixel 136 is very different from the intensities of the pixels around it and may be considered to be noise. (Generally, noise is information that does not belong to the surrounding environment.) The intensities of pixels 134 and 141 are also different from the neighboring pixels and may indicate a transition between the object and the background; thus, these pixels can be construed as representing the edges of the object.

Although we are discussing a discrete (digitized) signal, it may be transformed into a large number of sines and cosines with different amplitudes and frequencies that, if added together, will reconstruct the signal. As discussed earlier, slowly changing signals (such as small changes between succeeding pixel gray values) will require few sines and cosines in order to be reconstructed, and thus have low frequency content. On the other hand, quickly varying signals (such as large differences between pixel gray levels) will require many more frequencies to be reconstructed and thus have high frequency content. Both noises and edges are instances in which one pixel value is very different from the neighboring ones. Thus, noises and edges create the larger 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.

However, if a high-frequency signal is passed through a low-pass filter — a filter that allows lower frequencies to go through without much attenuation in amplitude, but that severely attenuates the amplitudes of the higher frequencies in the

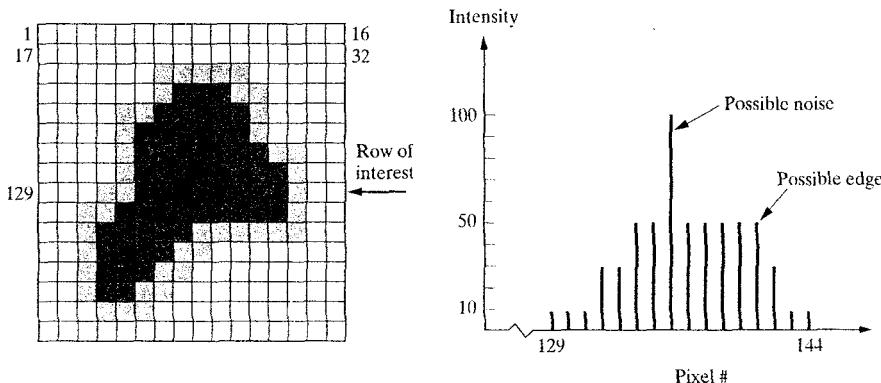


Figure 8.8 Noise and edge information in an intensity diagram of an image. The pixels with intensities that are much different from the intensities of neighboring pixels can be considered to be edges or noise.

signal—the filter will reduce the influence of all high frequencies, including the noises and edges. This means that, although a low-pass filter will reduce noises, it will also reduce the clarity of an image by attenuating the edges, thus softening the image throughout. A high-pass filter, on the other hand, will increase the apparent effect of higher frequencies by severely attenuating the low-frequency amplitudes. In such cases, noises and edges will be left alone, but slowly changing areas will disappear from the image.

To see how the Fourier transform can be applied in this case, let's look at the data of Figure 8.8 once again. The grayness level of the pixels of row 9 is repeated in Figure 8.9(a). A simple first-approximation Fourier transform of the gray values [4] was performed for the first four harmonic frequencies, and then the signal was reconstructed, as shown in Figure 8.9(b). Comparing the two graphs reveals that a digital, discrete signal can be reconstructed, even if its accuracy is dependent on the number of sines and cosines, as well as the method of integration, etc.

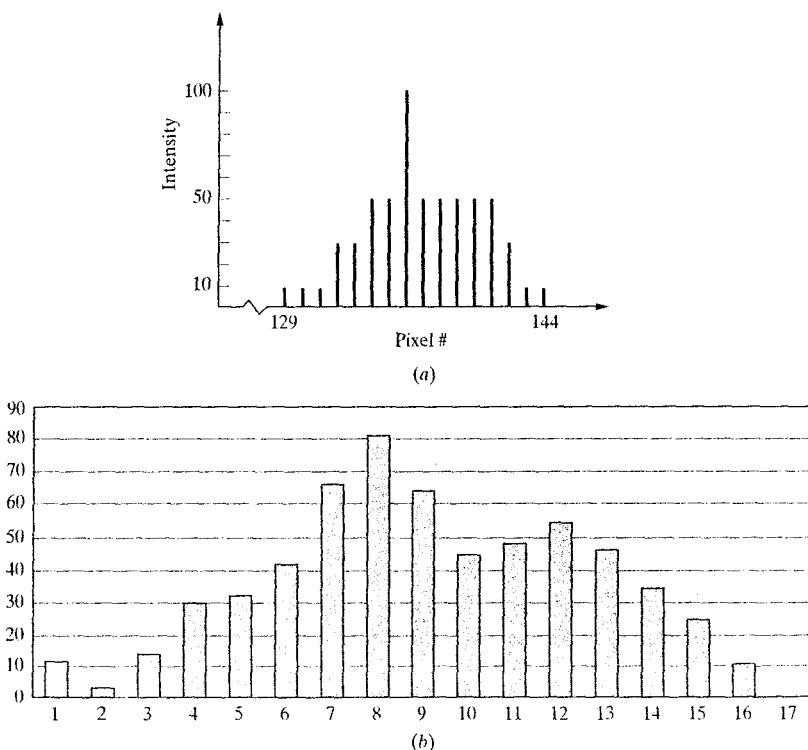


Figure 8.9 (a) Signal. (b) Discrete signal reconstructed from the Fourier transform of the signal in (a), using only four of the first frequencies in the spectrum.

8.10 SPATIAL-DOMAIN OPERATIONS: CONVOLUTION MASK

Spatial-domain processes access and operate on the information contained in an individual pixel. As a result, the image is directly affected by the operation. Most processes used in vision systems are in the spatial domain. One of the most popular and most common techniques in the spatial domain is the *convolution mask*, which can be adapted to many different tasks, from filtering to edge finding, to photography, and many more. We next examine the basic principles behind convolution masks, without referring to any particular type of mask. Later, we will adapt the convolution mask idea to different purposes.

Imagine that an image is composed of pixels, each with a particular gray level or color information, that collectively constitute the image. (Suppose that the gray level is not digitized into 0's and 1's, but the analog value is indicated.) As an example, let's say that the image in Figure 8.10 is part of a larger image with pixel values shown symbolically as A, B, C, \dots . Let's also assume that there is a 3×3 mask that has the values indicated by m_1, \dots, m_9 in its cells.

Applying the mask onto the image involves first superimposing the mask on the upper left corner of the image and taking the summation of the product of the value of each pixel and the corresponding mask value and then dividing the summation by a normalizing value. This yields

$$X = \frac{(A \times m_1 + B \times m_2 + C \times m_3 + E \times m_4 + F \times m_5 + G \times m_6 + I \times m_7 + J \times m_8 + K \times m_9)}{S}, \quad (8.2)$$

where

$$S = |m_1 + m_2 + m_3 + \dots + m_9| \quad (8.3)$$

is the normalizing value. However, if the summation is zero, a "1" is used.

The result X of this operation will be substituted for the value of the pixel in the center of the block that was superimposed. In this case, X will replace the value

A	B	C	D					
E	F	G	H					
I	J	K	L					
M	N	O	P					

m_1	m_2	m_3
m_4	m_5	m_6
m_7	m_8	m_9

Figure 8.10 A convolution mask superimposed on an image 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, which 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.

of $F(X \rightarrow F_{\text{new}})$. Usually, the substitution takes place in a new file in order to not alter the original file. The mask is then moved one pixel to the right, and the same operation is repeated for a new value of X , which will replace G in a new file as follows:

$$X = G_{\text{new}} = \frac{(B \times m_1 + C \times m_2 + D \times m_3 + F \times m_4 + G \times m_5 + H \times m_6 + J \times m_7 + K \times m_8 + L \times m_9)}{S}$$

Next, the mask is moved over one more pixel, and the operation is repeated until all the pixels in the row are changed. Then the operation continues with the subsequent rows until the image is completely affected. The resulting image will show characteristics that may be slightly or very severely affected by the operation, depending on the m values in the mask. The first and last rows and columns are not affected and are usually ignored. Some systems insert zeros for the first and last rows and columns.

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, the value of the pixel $(I_{x,y})_{\text{new}}$ at the center of a block can be calculated by

$$(I_{x,y})_{\text{new}} = \frac{\sum_{i=1}^n \sum_{j=1}^n M_{i,j} \times I_{\left[\left(x - \left(\frac{n+1}{2}\right) + i\right), \left(y - \left(\frac{n+1}{2}\right) + j\right)\right]}}{S}, \quad (8.4)$$

where

$$S = \left| \sum_{i=1}^n \sum_{j=1}^n M_{i,j} \right| \quad \text{if the summation } \neq 0 \quad (8.5)$$

and

$$S = 1 \quad \text{if the summation} = 0.$$

Note that the normalizing or scaling factor S is arbitrary and is used to prevent saturation of the image. As a result, the user can always adjust this number to get the best image without saturation.

Example 8.1

Consider the pixels of an image, with values as shown in Figure 8.11, as well as a convolution mask with the given values. Calculate the new values of the given pixels.

Solution We will substitute zeros for the first and last columns and rows, because they are not affected by convolution. For the remaining pixels, we will superimpose the mask on the remaining cells of the image and will use Equations (8.2) and (8.3) to calculate new pixel values, as shown in Figure 8.12(a), with the result as indicated in

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 8.11 An example of a convolution mask.

2,2

5x0	6x0	2x1	8
3x1	3x1	5x1	6
4x1	3x0	2x0	6
8	6	5	9

0	0	1
1	1	1
1	0	0

2,3

5	6x0	2x0	8x1
3	3x1	5x1	6x1
4	3x1	2x0	6x0
8	6	5	9

0	0	1
1	1	1
1	0	0

3,2

5	6	2	8
3x0	3x0	5x1	6
4x1	3x1	2x1	6
8x1	6x0	5x0	9

0	0	1
1	1	1
1	0	0

3,3

5	6	2	8
3	3x0	5x0	6x1
4	3x1	2x1	6x1
8	6x1	5x0	9x0

0	0	1
1	1	1
1	0	0

(b)

0	0	0	0
0	3.4	5	0
0	4.4	4.6	0
0	0	0	0

(b)

Figure 8.12 (a) Superimposing the mask onto the cells of the image. (b) The result of the operation.

Figure 8.12(b). Superimposing the mask on the image for each remaining element, we have the following equations:

$$2,2: \quad 5(0) + 6(0) + 2(1) + 3(1) + 3(1) + 5(1) + 4(1) + 3(0) + 2(0)/5 = 3.4;$$

$$2,3: \quad 6(0) + 2(0) + 8(1) + 3(1) + 5(1) + 6(1) + 3(1) + 2(0) + 6(0)/5 = 5;$$

$$3,2: \quad 3(0) + 3(0) + 5(1) + 4(1) + 3(1) + 2(1) + 8(1) + 6(0) + 5(0)/5 = 4.4;$$

$$3,3: \quad 3(0) + 5(0) + 6(1) + 3(1) + 2(1) + 6(1) + 6(1) + 5(0) + 9(0)/5 = 4.6.$$

We next consider common routines and techniques that are used in image processing and image analysis.

8.11 SAMPLING AND QUANTIZATION

To be useful in image processing, the image must be digitized both spatially as well as in amplitude. Spatial digitization is the process that was mentioned in Section 8.5.2, wherein the intensities of light at each pixel location are read. The more pixels that are present and individually read, the better is the resolution of the camera and the image. This technique is called *sampling*, as the light intensities are sampled at equally spaced intervals. A larger sampling rate will create a larger number of pixel data and thus better resolution.

Figure 8.13 shows the same image sampled at (a) 432×576 , (b) 108×144 , (c) 54×72 , and (d) 27×36 pixels. As the sampling rate decreases, the clarity of the image is lost. The voltage or charge read at each pixel value is an analog value and must also be digitized. Digitization of the light intensity values at each pixel location is called *quantization*. Depending on the number of bits used, the resolution 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: “off” or “on,” or, alternatively, 0 or 1 (called a *binary image*). For quantization with an 8-bit ADC, the maximum number of gray levels will be 256. Thus, the image will have 256 different gray levels in it. Quantization and sampling resolutions are completely independent of each other. For example, a high-resolution image may be converted into a binary image, and thus the quantization is only into two digits (0 and 1, or black and white, or “off” and “on”). Still, the same image may be quantized into 8 bits, which can yield a spectrum of 256 different shades of gray.



Figure 8.13 Effect of different sampling rates on an image at (a) 432×576 , (b) 108×144 , (c) 54×72 , and (d) 27×36 pixels. As the resolution decreases, the clarity of the image diminishes accordingly.

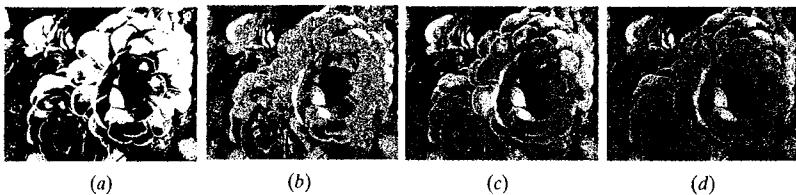


Figure 8.14 An image quantized at 2, 4, 8, and 44 gray levels. As the quantization resolution increases, the image becomes smoother.

Figure 8.14 shows the same image quantized at (a) 2 levels, (b) 4 levels, (c) 8 levels, and (d) the original 44 levels.

The sampled light at a pixel, when quantized, will yield a string of 0's and 1's 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 and the memory needed for each digitized sample. The larger the image size, the resolution of the image, or the number of gray levels, the larger is the memory requirement.

Example 8.2

Consider an image with 256 by 256 pixels. The total number of pixels in the image will be $256 \times 256 = 65,536$. If the image is binary, it will require 1 bit to record each pixel as 0 or 1. Thus, the total memory needed to record the image will be 65,536 bits, or, with 8 bits to a byte, 8,192 bytes. If each pixel were to be digitized at the rate of 8 bits for 256 shades of gray, it would require $65,536 \times 8 = 524,288$ bits, or 65,536 bytes. If the image were in color, it would require 65,536 bytes for each of the three colors of red, green, and blue. For a color video clip changing at the rate of 30 images per second, the memory requirement will be $65,536 \times 3 \times 30 = 5,898,240$ bytes per second. Of course, this is only the memory requirement for recording the image pixels and does not include index information and other bookkeeping requirements.

8.12 SAMPLING THEOREM

Can you tell what the image in Figure 8.15 represents? Of course, since it is a very low resolution 16×16 image, it is difficult to guess what the object is. This simple illustration signifies the relationship between the sampling rate and the information obtained from it. To understand the relationship, we will discuss some fundamental issues connected with sampling.

Consider a simple sinusoidal signal with frequency f_s as shown in Figure 8.16(a). Suppose that the signal is sampled at the rate of f_s . This means that the sampling circuit will read the amplitude of the signal with a frequency of f_s . The arrows in Figure 8.16(b) show the corresponding sampled amplitudes.

Now suppose that we want to use the sampled data to reconstruct the signal. (Doing this would be similar to sampling a sound source such as a CD player and then trying to reconstruct the sound signal from the sampled data through a speaker.) One possibility would be that, by chance, the same signal might be reconstructed. However, as you see in Figure 8.17, it is quite possible that, from the same data,

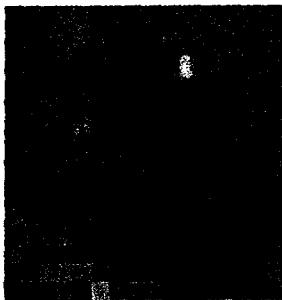


Figure 8.15 A low-resolution (16×16) image.

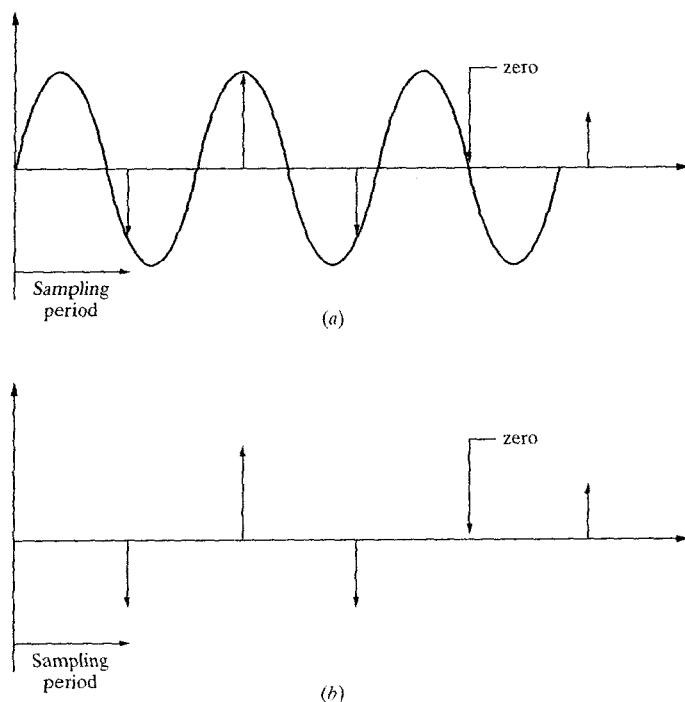


Figure 8.16 (a) Sinusoidal signal with a frequency of f . (b) Amplitudes sampled at the rate of f_s .

another signal may be reconstructed which is completely different from the original signal. Both signals are valid, and in fact, many other signals can be valid as well and might be reconstructed from the sampled data. This loss of information is called *aliasing* of the sampled data, and it can be a serious problem.

In order to prevent aliasing, according to what is referred to as the *sampling theorem*, the sampling frequency must be at least twice as large as the largest fre-

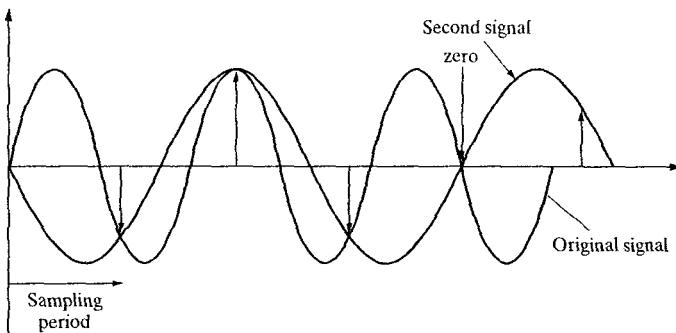


Figure 8.17 Reconstruction of signals from sampled data. More than one signal may be reconstructed from the same sampled data.

frequency present in the signal. In that case, one can reconstruct the original signal without aliasing. The highest frequency present in the signal can be determined from the frequency spectrum of the signal. Using the Fourier transform, one finds that the frequency spectrum of a signal will contain many frequencies. However, as we have seen, the higher frequencies may have smaller amplitudes. One 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 system's total representation. In that case, the sampling rate of the signal must be at least twice as large as this maximum frequency. In practice, the sampling rate is generally chosen to be even larger, to further ensure that aliasing of the signal will not occur. Frequencies four to five times as large as the maximum frequency are common. As an example, consider a CD player. Theoretically, human ears can hear frequencies of up to about 20,000 Hz. If the CD player is to be able to reconstruct the digitized sampled music, the sampling rate of the laser sensor 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 the example of Figure 8.18, the sampling rate is lower than the higher frequencies of the signal. Although the lower frequencies of the signal are reconstructed, the signal will not have the higher frequencies of the original signal. The same may happen to any signal, including audio and video signals.

For images, too, if the sampling rate (which translates into the resolution of the image) is low, the sampled data may not have all the necessary detail, the information in the image is lost, and the image cannot be reconstructed to match the original. The image in Figure 8.15 is sampled at a very low rate, and the information in it is lost. This is why you cannot tell what the image is. However, if the sampling rate is increased, there will be a time when there will be enough information to be able to recognize the image. The still higher resolutions or sampling rates will transfer more information, and thus, increasingly more detail can be recognized. Figure 8.19 is the same image as in Figure 8.15, but at 2, 4, and 16 times higher resolutions. Now suppose that you need to recognize the difference between a bolt and a

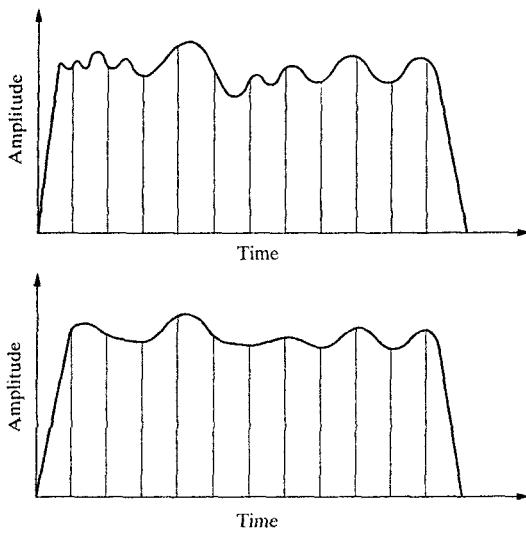


Figure 8.18 The original signal in (a) is sampled at a sampling rate that is lower than the higher frequencies of the signal. The reconstructed signal in (b) will not have the higher frequencies of the original signal.

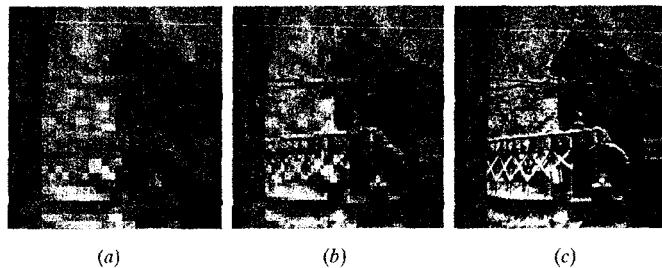


Figure 8.19 The image of Figure 8.15, presented at higher resolutions of (a) 32×32 , (b) 64×64 , and (c) 256×256 .

nut in a vision system in order to direct a robot to pick up the parts. Because the information representing a bolt is very different from that representing a nut, low-resolution images will still enable you to determine what the part is. However, in order to recognize the license plate number of a car while moving in traffic, one would need to have a high-resolution image to extract enough information about the details, such as the numbers on the license plate.

8.13 IMAGE-PROCESSING TECHNIQUES

As was mentioned earlier, image-processing techniques are used to enhance, improve, or otherwise alter an image and to prepare it for image analysis. Usually, during image processing information is not extracted from the image. The intention is to remove faults, trivial information, or information that may be important, but not useful, and to improve the image. As an example, suppose that 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, etc.) can be determined. Again, consider an image that is corrupted by direct lighting that is reflected back, 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 an image of a section of a city that is fully detailed, with streets, cars, shadows, etc. 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 subprocesses, including histogram analysis, thresholding, masking, edge detection, segmentation, region growing, and modeling, among others. In the next few sections, we will study some of these processes and their application.

8.14 HISTOGRAM 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 when an image is to be transformed into binary values. It can also be used to decide whether there are any prevalent gray levels in an image. For instance, suppose a systematic source of noise in an image causes many pixels to have one “noisy” gray level. Then a histogram can be used to determine what the noisy gray level is in order to attempt to remove or neutralize the noise.

Now suppose that an image has all its pixel gray levels clustered between two relatively close values, as in Figure 8.20(a). In this image, all pixel gray values are between 120 and 180 gray levels, at four-unit intervals. (The image is quantized at 16 distinct levels between 0 and 256.) Figure 8.20(c) is the histogram of the image, and clearly, all pixel gray levels are between 120 and 180, a relatively low range. 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 16 gray levels present in the image are spread out between 0 and 255 gray levels, at intervals of 17 units, instead of the present 120–180 gray levels at intervals of 4 units. Then, due to the equalization, the image is vastly improved, as shown in Figure 8.20(b), with its corresponding histogram in Figure 8.20(d). Notice that the number of pixels at each gray level is exactly the same in both cases, but that only the gray levels are spread out. The grayness values are given in Table 8.1.

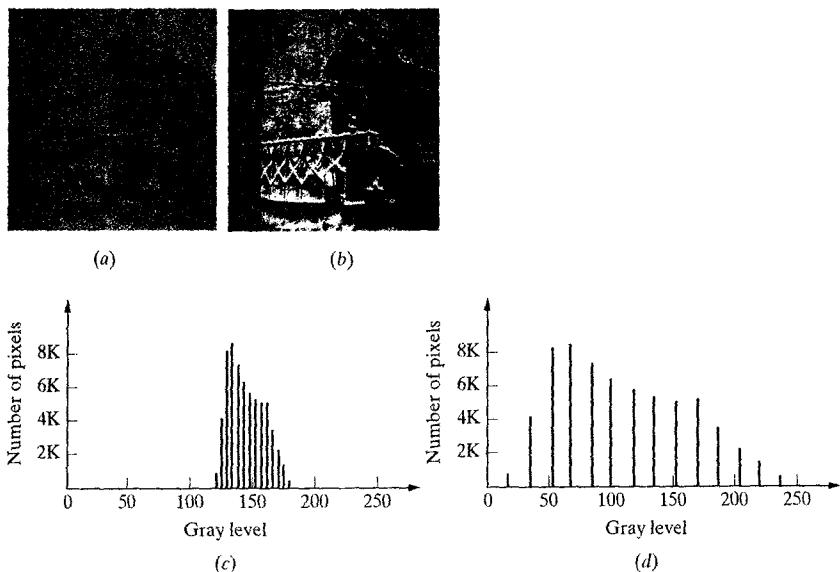


Figure 8.20 Effect of histogram equalization in improving an image.

TABLE 8.1 THE ACTUAL GRAYNESS VALUES AND NUMBER OF PIXELS FOR THE IMAGES IN FIGURES 8.20 (A) AND (B)

Levels	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
No. of Pixels	0	750	5,223	8,147	8,584	7,769	6,419	5,839	5,392	5,179	5,185	3,451	2,078	1,692	341	0
For (a)	0	17	34	51	68	85	102	119	136	153	170	187	204	221	238	256
For (b)	120	124	128	132	136	140	144	148	152	156	160	164	168	172	176	180

8.15 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 value with the threshold, and then assigning the pixel to the different portions or levels, depending on whether the pixel's grayness level is below the threshold (off or zero, or not belonging) or above the threshold (on or 1, or belonging). Thresholding can be performed either at a single level or at multiple levels, in which the image is processed by dividing it into "layers," each with a selected threshold. To aid in choosing an appropriate threshold, many different techniques have been suggested, ranging from simple routines for binary images to sophisticated techniques for com-

plicated images. Early routines used for a binary image had the object lighted and the background completely dark. This condition can be achieved in controlled lighting in industrial situations, but may not be possible in other environments. In binary images, the pixels are either on or off, and thus, choosing a threshold is simple and straightforward. In certain other situations, the image will have multiple gray levels, and its histogram will exhibit a bimodal distribution. In this case, the valley is chosen as the threshold value. More advanced techniques use statistical information and distribution characteristics of the image pixels to develop a thresholding value. As the thresholding value changes, so does the image. Figure 8.21 shows an original image with 256 gray levels and the result of thresholding at grayness levels of 100 and 150.

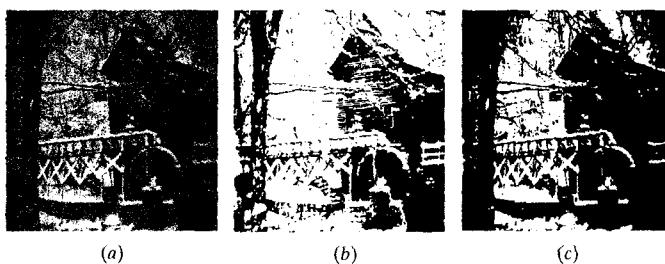


Figure 8.21 Thresholding an image with 256 gray levels at values of (b) 100 and (c) 150.

Thresholding is used in many operations, such as transforming an image into binary values, filtering operations, masking, and edge detection.

8.16 CONNECTIVITY

Sometimes we need to decide whether neighboring pixels are somehow “connected” or related to each other. Connectivity establishes whether they have the same properties, such as being of the same region, coming from the same object, having a similar texture, etc. To establish connectivity of neighboring pixels, we first have to decide upon a connectivity path. For example, we need to decide whether only pixels that are on the same column and row are connected or whether diagonally situated pixels are also accepted as being connected.

There are three fundamental connectivity paths for two-dimensional image processing and analysis: +4- or $\times 4$ -connectivity, H6 or V6 connectivity, and 8-connectivity. In three dimensions, connectivity between voxels (volume cells) can range from 6 to 26. The following terms are defined with respect to Figure 8.22.

+4-connectivity applies when a pixel p 's relationship is analyzed only with respect to the four pixels immediately above, below, to the left, and to the right of p (namely, b, g, d , and e).

a	b	c
d	p	e
f	g	h

Figure 8.22 Neighborhood connectivity of pixels.

$\times 4$ -connectivity applies when a pixel p 's relationship is analyzed only with respect to the four pixels immediately across from it diagonally on 4 sides (a , c , f , and h).

For a pixel $p(x,y)$ the relevant pixels are as follows:

- for $+4$ -connectivity: $(x+1, y), (x-1, y), (x, y+1), (x, y-1); \quad (8.6)$

- for $\times 4$ -connectivity: $(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1). \quad (8.7)$

H6-connectivity applies when a pixel p 's relationship is analyzed only with respect to the six neighboring pixels on the two rows immediately above and below p (a , b , c , f , g , h).

V6-connectivity applies when a pixel p 's relationship is analyzed only with respect to the six neighboring pixels on the two columns immediately to the right and to the left of p (a , d , f , c , e , h).

For a pixel $p(x,y)$, the relevant pixels are as follows:

- for H6-connectivity: $(x-1, y+1), (x, y+1), (x+1, y+1), (x-1, y-1), (x, y-1), (x+1, y-1); \quad (8.8)$

- for V6-connectivity: $(x-1, y+1), (x-1, y), (x-1, y-1), (x+1, y+1), (x+1, y), (x+1, y-1). \quad (8.9)$

8-connectivity applies when a pixel p 's relationship is analyzed with respect to all eight pixels surrounding it (a , b , c , d , e , f , g , h).

For a general pixel $p(x,y)$, the relevant pixels are as follows:

$$(x-1, y-1), (x, y-1), (x+1, y-1), (x-1, y), (x+1, y), (x-1, y+1), (x, y+1), (x+1, y+1). \quad (8.10)$$

So far, we have studied some general issues and fundamental techniques that are used in image processing and analysis. Next, we will discuss particular techniques that are used for specific applications.

Example 8.3

In the image of Figure 8.23, starting with pixel (4, 3), find all succeeding pixels that can be considered as connected to each other based on $+4$ -, $\times 4$ -, H6-, V6-, and 8-connectivity rules.

Solution Figure 8.24 shows the results of the five connectivity searches. For each search, take one pixel, find all others that are connected to it based on the rule you are working with, and then search the pixels that were found to be connected to the previ-

1,1	1,2	1,3	1,4	1,5	1,6
2,1	2,2	2,3	2,4	2,5	2,6
3,1	3,2	3,3	3,4	3,5	3,6
4,1	4,2	4,3	4,4	4,5	4,6
5,1	5,2	5,3	5,4	5,5	5,6
6,1	6,2	6,3	6,4	6,5	6,6

Figure 8.23 The image for Example 8.3.

1,1	1,2	1,3	1,4	1,5	1,6
2,1	2,2	2,3	2,4	2,5	2,6
3,1	3,2	3,3	3,4	3,5	3,6
4,1	4,2	4,3	4,4	4,5	4,6
5,1	5,2	5,3	5,4	5,5	5,6
6,1	6,2	6,3	6,4	6,5	6,6

1,1	1,2	1,3	1,4	1,5	1,6
2,1	2,2	2,3	2,4	2,5	2,6
3,1	3,2	3,3	3,4	3,5	3,6
4,1	4,2	4,3	4,4	4,5	4,6
5,1	5,2	5,3	5,4	5,5	5,6
6,1	6,2	6,3	6,4	6,5	6,6

1,1	1,2	1,3	1,4	1,5	1,6
2,1	2,2	2,3	2,4	2,5	2,6
3,1	3,2	3,3	3,4	3,5	3,6
4,1	4,2	4,3	4,4	4,5	4,6
5,1	5,2	5,3	5,4	5,5	5,6
6,1	6,2	6,3	6,4	6,5	6,6

+4

x 4

H6

1,1	1,2	1,3	1,4	1,5	1,6
2,1	2,2	2,3	2,4	2,5	2,6
3,1	3,2	3,3	3,4	3,5	3,6
4,1	4,2	4,3	4,4	4,5	4,6
5,1	5,2	5,3	5,4	5,5	5,6
6,1	6,2	6,3	6,4	6,5	6,6

1,1	1,2	1,3	1,4	1,5	1,6
2,1	2,2	2,3	2,4	2,5	2,6
3,1	3,2	3,3	3,4	3,5	3,6
4,1	4,2	4,3	4,4	4,5	4,6
5,1	5,2	5,3	5,4	5,5	5,6
6,1	6,2	6,3	6,4	6,5	6,6

V6

8

Figure 8.24 The results of the connectivity searches for Example 8.3.

ous ones for additional connected pixels, until you are done. All of the remaining pixels will not be connected. We will use the same rules later for other purposes, such as region growing.

8.17 NOISE REDUCTION

Like other signal-processing mediums, vision systems contain noises. Some noises are systematic and come from dirty lenses, faulty electronic components, bad mem-

ory chips, and low resolution. Others are random and are caused by environmental effects or bad lighting. The net effect is a corrupted image that needs to be pre-processed to reduce or eliminate the noise. In addition, sometimes images are not of good quality, due to both hardware and software inadequacies; thus, they have to be enhanced and improved before other analyses can be performed on them. On the hardware level, in one attempt [5], an on-chip correction scheme was devised for defective pixels in an image sensor. In this scheme, readouts from nearest neighbors were substituted for defective pixels that had been identified. However, in general, software schemes are used for most filtering operations.

Filtering techniques are divided into two categories. *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.

8.17.1 Convolution Masks

As was mentioned in Section 8.10, a mask may be used for many different purposes, including filtering operations and noise reduction. In Section 8.9, it was also mentioned that noise, as well as edges, produces higher frequencies in the spectrum of a signal. It is possible to create masks that behave like a low-pass filter, such that the higher frequencies of an image are attenuated while the lower frequencies are not changed very much. Thereby, the noise is reduced.

Neighborhood Averaging

Neighborhood averaging can be used to reduce noise in an image, but it also reduces the sharpness of the image. Consider the 3×3 mask shown in Figure 8.25 together with its corresponding values, as well as a portion of an imaginary image with its gray levels indicated. As you can see, all the pixels but one are at a gray value of 20. The pixel with a gray level of 100 may be considered to be noise, since it is dif-

20	20	20	20					
20	100	20	20					
20	20	20	20					

1	1	1
1	1	1
1	1	1

Figure 8.25 Neighborhood-averaging mask.

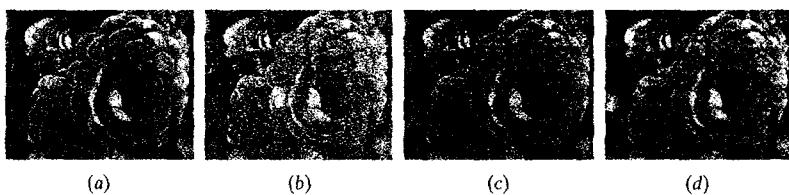


Figure 8.26 Neighborhood averaging of an image.

<table border="1"> <tbody> <tr><td>1</td><td>4</td><td>6</td><td>4</td><td>1</td></tr> <tr><td>4</td><td>16</td><td>24</td><td>16</td><td>4</td></tr> <tr><td>6</td><td>24</td><td>36</td><td>24</td><td>6</td></tr> <tr><td>4</td><td>16</td><td>24</td><td>16</td><td>4</td></tr> <tr><td>1</td><td>4</td><td>6</td><td>4</td><td>1</td></tr> </tbody> </table> 5×5	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	<table border="1"> <tbody> <tr><td>1</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>4</td><td>2</td></tr> <tr><td>1</td><td>2</td><td>1</td></tr> </tbody> </table> 3×3	1	2	1	2	4	2	1	2	1
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																															
1	2	1																																	
2	4	2																																	
1	2	1																																	

Figure 8.27 5×5 and 3×3 Gaussian averaging filters.

ferent from the pixels around it. Applying the mask over the corner of the image, with a normalizing value of 9 (the sum of all the values in the mask), yields:

$$X = \frac{(20 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1 + 100 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1)}{9} \\ = 29.$$

As a result of applying the mask on the indicated corner, the pixel with the value of 100 changes to 29, and the large difference between the noisy pixel and the surrounding pixels (100 vs. 20) becomes much smaller (29 vs. 20), thus reducing the noise. With this characteristic, the mask acts as a low-pass filter. Notice that the operation will introduce new gray levels into the image (29) and thus will change its histogram. Similarly, this averaging low-pass filter will also reduce the sharpness of edges, making the resulting image softer and less focused. Figure 8.26 shows an original image (a), a corrupted image with noise (b), the image after a 3×3 averaging filter application (c), and the image after a 5×5 averaging filter application (d). As you can see, the 5×5 filter works even better than the 3×3 filter, but requires a bit more processing.

There are other averaging filters, such as the Gaussian averaging filter (also called the mild isotropic low-pass filter), which is shown in Figure 8.27. This filter will similarly improve an image, but with a slightly different result.

8.17.2 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 be halted. As a result, in addition to being time consuming, the technique is not suitable for operations that are dynamic and change rapidly. Image averaging is more effective with an increased number of images. It is fundamentally useful for noise that is random; if the noise is systematic, its effect on the image will be exactly the same for all multiple images, and as a result, averaging will not reduce the noise. 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 from averaging because the summation of random noises will be zero; that is,

$$A(x,y) = I(x,y) + N(x,y)$$

$$\frac{\sum_n A(x,y)}{n} = \frac{\sum_n I(x,y) + N(x,y)}{n} = \frac{\sum_n I(x,y)}{n} + \frac{\sum_n N(x,y)}{n} = I(x,y). \quad (8.11)$$

Although image averaging reduces random noise, unlike neighborhood averaging, it does not blur the image or reduce its focus.

8.17.3 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.

8.17.4 Median Filters

One of the main problems in using neighborhood averaging is that, along with removing noises, the filter will blur edges. A variation of this technique is to use a median filter, in which the value of the pixel is replaced by the median of the values of the pixels in a mask around the given pixel (the given pixel plus the eight surrounding pixels), sorted in ascending order. A median is the value such that 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 is independent of the value of any single pixel in the set, the median filter will be much stronger in eliminating spikelike noises without blurring the object or decreasing the overall sharpness of the image. Suppose that we apply a median filter to the image in Figure 8.25. Then sorted values, in ascending order, will be 20, 20, 20, 20, 20, 20, 20, 20, 100. The median is thus the fifth 20 in the sequence. Replacing the center pixel's value with 20 will completely eliminate the noise. Of course, noises are not always this easily removed, but the 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.

Median filters tend to make the image grainy, especially if applied more than once. Consider the image in Figure 8.28(a). The gray values, in ascending order, are 1, 2, 3, 4, 5, 6, 7, 8, 9. The middle value is 5, resulting in the image in (b). Observe that the image has become grainy because the pixel sets with similar values appear longer (as in 5 and 5).

2	1	3			
8	9	4			
7	5	6			

(a)

2	1	3			
8	5	4			
7	5	6			

(b)

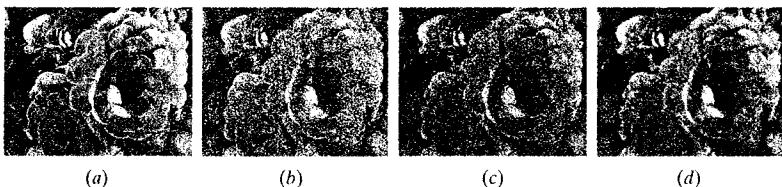
Figure 8.28 Application of a median filter.**Figure 8.29** (a) Original image. (b) The same image corrupted with a random Gaussian noise. (c) The image improved by a 3×3 median filter. (d) The same image improved with a 7×7 median filter.

Figure 8.29 shows an original image (a), the image corrupted with random Gaussian noise (b), and the image improved with a 3×3 median filter (c) and a 7×7 median filter (d).

8.18 EDGE DETECTION

Edge detection is a general name for a class of routines and techniques that operate on an image and result in a line drawing of the image. The lines represent changes in values such as cross sections of planes, intersections of planes, textures, lines, and colors, as well as differences in shading and textures. Some techniques are mathematically oriented, some are heuristic, and some are descriptive. All 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 to be stored, is much simpler to be processed, and saves in computation and storage costs. Edge detection is also necessary in subsequent processes, such as segmentation and object recognition. Without edge detection, it may be impossible to find overlapping parts, to calculate features such as a diameter and an area, or to determine parts by region growing. Different techniques of edge

detection yield slightly different results; thus, they should be chosen carefully and used wisely.

As was mentioned earlier, like noise, edges create higher frequencies in the spectrum and hence 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, thereby separating the noises and edges from the rest of the image. Consider the image and the mask (called a Laplacian1 filter) in Figure 8.30. The mask has negative numbers in it. Applying the mask to the image at the corner will result in

$$X = \frac{(20 \times -1 + 20 \times 0 + 20 \times -1 + 20 \times 0 + 100 \times 4 + 20 \times 0 + 20 \times -1 + 20 \times 0 + 20 \times -1)}{1} \\ = 320$$

The normalizing factor is 1, which results in the value of 100 being replaced with 320, accentuating the original difference (from 100 vs. 20 to 320 vs. 20). Since this mask accentuates differences (higher frequencies), it is a high-pass filter, which also means that the noise and edges of objects in images will be shown more effectively. As a result, the mask acts as an edge detector. Some high-pass filters can act as an image sharpener. Figure 8.31 shows some other high-pass filters.

Still other masks have the effect of differentiating an image through the use of gradients. In this case, the horizontal and vertical gradients between neighboring

20	20	20	20		
20	100	20	20		
20	20	20	20		

-1	0	-1
0	4	0
-1	0	-1

Figure 8.30 A typical high-pass edge detector mask (Laplacian1).

-1	-1	-1
-1	8	-1
-1	-1	-1

0	-1	0
-1	6	-1
0	-1	0

0	-1	0
-1	5	-1
0	-1	0

Laplacian2

Sharpen, low

Sharpen, medium

Figure 8.31 Other high-pass filters.

pixels are calculated and squared, and then the square root of the sum is found. Mathematically,

$$\nabla f = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}. \quad (8.12)$$

Equation (8.12) is equivalent to calculating the absolute value of the differences between pixel intensities. The three masks [6,7,8,9,10] shown in Figure 8.32 and called the Sobel, Roberts, and Prewitt edge detectors, effectively do the same gradient differentiation with somewhat different results and are very common. When they are applied to an image, the two pairs of masks calculate the gradients in the x and y directions, are added, and then are compared with a threshold. Figure 8.33 is an original image (a) with its edges detected by a Laplacian1 (b), Laplacian2 (c), Sobel (d), and Robert's (e) edge detectors. The result for other images may be different 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 simple routines that are easy to implement and that yield continuous edges can be used for binary images. In one example [11], a search technique, dubbed left-right (L-R) in this book, is used that can quickly and efficiently detect edges in binary images of single objects which look like a blob. Imagine a binary image such as that shown in Figure 8.34. Suppose that gray pixels are “on” (or represent the object) and white pixels are “off” (or represent the background). Assume that a pointer is moving from one pixel to another, in any direction (up, down, right, or left). Anytime the pointer reaches an “on” pixel, it will turn left. Anytime it reaches an “off” pixel, it will turn right. Of course, as shown, depending on the direction of the pointer, “left” and “right” might mean different directions as well.

$\begin{array}{ c c } \hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$	$\begin{array}{ c c } \hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$
(a) Sobel	(b) Roberts				(c) Prewitt

Figure 8.32 The Sobel, Roberts, and Prewitt edge detectors.

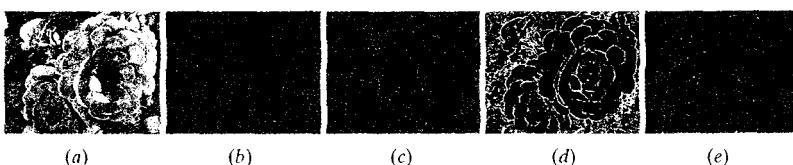


Figure 8.33 An image (a) and its edges from Laplacian1 (b), Laplacian2 (c), Sobel (d), and Robert's (e) edge detectors.

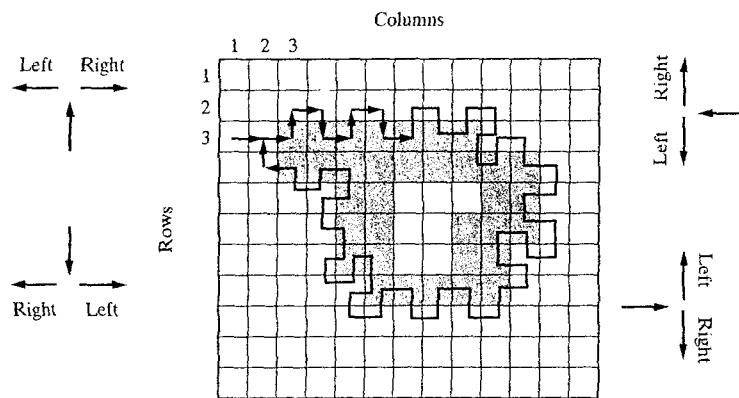


Figure 8.34 Left-right search technique for edge detection [11].

Starting at pixel 1,1, moving to 1,2, to the end of the row and then onto row 2, row 3, etc., 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 will be pixels 3,3; 3,4; 3,5; 3,6; . . . , 3,9; 4,9; 4,10; 4,11;

Masks may also be used to intentionally emphasize some characteristic of the image. For example, a mask may be designed to emphasize horizontal lines, vertical lines, or diagonal lines. Figure 8.35 shows three such masks. Figure 8.36 shows an

<table border="1"><tr><td>3</td><td>-6</td><td>3</td></tr><tr><td>3</td><td>-6</td><td>3</td></tr><tr><td>3</td><td>-6</td><td>3</td></tr></table>	3	-6	3	3	-6	3	3	-6	3	<table border="1"><tr><td>3</td><td>3</td><td>3</td></tr><tr><td>-6</td><td>-6</td><td>-6</td></tr><tr><td>3</td><td>3</td><td>3</td></tr></table>	3	3	3	-6	-6	-6	3	3	3	<table border="1"><tr><td>3</td><td>3</td><td>-6</td></tr><tr><td>3</td><td>-6</td><td>3</td></tr><tr><td>-6</td><td>3</td><td>3</td></tr></table>	3	3	-6	3	-6	3	-6	3	3
3	-6	3																											
3	-6	3																											
3	-6	3																											
3	3	3																											
-6	-6	-6																											
3	3	3																											
3	3	-6																											
3	-6	3																											
-6	3	3																											
Vertical mask	Horizontal mask	Diagonal mask																											

Figure 8.35 Masks emphasizing the vertical, horizontal, and diagonal lines of an image.

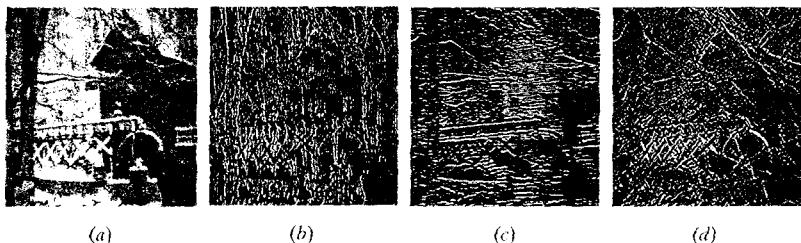


Figure 8.36 An original image (a) with effects of vertical mask (b), horizontal mask (c), and diagonal mask (d).

original image (a), along with the effects of a vertical mask (b), a horizontal mask (c), and a diagonal mask (d).

8.19 HOUGH TRANSFORM

As you have probably noticed, in most edge detection techniques, the resulting edges are not continuous. However, in many applications, continuous edges are either necessary or preferred. For example, as we will see later, in region growing, edges that define an area or a region must be continuous and complete before a region-growing routine can detect the region and label it. In addition, 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 [12] is a technique used to determine the geometric relationship between different pixels on a line, including the slope of the line. For example, one can determine whether a cluster of points is on a straight line. This kind of determination aids in further developing an image in preparation for object recognition, since it transforms individual pixels into recognizable forms. The Hough transform is based on transforming the image space into an (r, θ) space, which in turn is based on the fact that an infinite number of straight lines go through any point in a plane. The normal to any one of these lines through the origin will have an angle of θ with respect to the x -axis and will be at a distance of r from the origin. The transformation into the (r, θ) plane (also called the Hough plane) showing these values is the Hough transform. Alternatively, a line in the xy -plane, with slope m and intercept c , can be transformed into a Hough plane of m, c , with x and y as its slope and intercept. Thus, a line in the xy -plane with a particular slope and intercept will transform into a point in the Hough plane. All lines through a point will transform into a single line in the Hough plane. If a group of points are collinear, their Hough transforms will all intersect. By examining these properties in a particular case, it can be determined whether a cluster of pixels is on a straight line. Hough transforms can also be used to determine the angle or orientation of a line, whereupon the orientation of an object in a plane can be determined by calculating the orientation of a particular line in the object.

To make this clearer, let's consider a plane xy (Figure 8.37) with a line in it. The line can be described by its slope (m) and intercept (c) as

$$y = mx + c. \quad (8.13)$$

Equation (8.13) can also be written in terms of m and c as

$$c = -xm + y, \quad (8.14)$$

where, in the mc -plane, the x and y will be the slope and the intercept.

Clearly, the line given by Equation (8.13) with m and c will be shown as a single point A in the mc - (Hough) plane. This is because all points on the line have the same slope and intercept m, c , and all produce the same location in that 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.

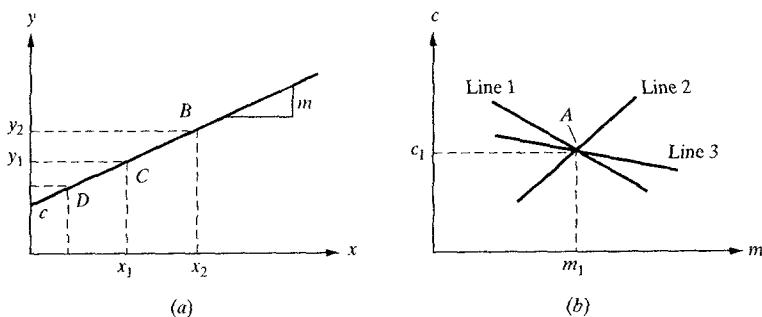


Figure 8.37 Hough transform.

Now consider the Hough plane in Figure 8.37 (b). Two lines intersect at point A . From Equation (8.14), Line 1 has a slope and intercept of x_1 and y_1 , which can be shown as point C in the xy -plane. Similarly, Line 2 can be shown as point B in the xy -plane. Thus, each line in the mc -plane transforms into a point in the xy -plane. This means that if two lines intersect in the mc -plane, they will form a line in the xy -plane. If a third line intersects at the same point, it will have to be on the same line in xy -plane. As a result, if a number of points in the xy -plane are on the same line, they correspond to intersecting lines in the mc -plane. This property can be used to determine whether a number of points are all on the same line. If one takes five points in the xy -plane, for example, so long as they all are on the same line, their corresponding lines will all intersect at the same point in the mc -plane. Of course, using the slope and intercept of the line, one can add points to the line to make it continuous or to close a shape.

Example 8.4

- Following are the coordinates of five points:

y	x
3	1
2	2
1.5	3
1	4
0	5

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 are on a line. So we will look for at least three points that will be on the same line. Looking at the graph of the points, we see that it is very easy — even trivial — 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, for either a computer or a human being, to tell which points are on the

same line and which are not. We will use a Hough transform to determine which points fall on the same line. The following table summarizes the lines formed in the mc -plane that correspond to the points shown in the xy -plane:

y	x	xy	mc
3	1	$3 = m1 + c$	$c = -1m + 3$
2	2	$2 = m2 + c$	$c = -2m + 2$
1.5	3	$1.5 = m3 + c$	$c = -3m + 1.5$
1	4	$1 = m4 + c$	$c = -4m + 1$
0	5	$0 = m5 + c$	$c = -5m + 0$

Figure 8.38 shows the five corresponding lines drawn in the mc -plane. Three of the lines are intersecting, while the other two are not. The latter lines correspond to points 1,3 and 5,0. The slope and intercept of the line representing the points that are on the same line are -0.5 and 3, respectively. Once again, these lines intersect at other points as well, indicating that any two points form a line. This shows how the Hough transform can be cluttered with numerous intersecting lines. Determining which lines are intersecting is the main issue in Hough transform analysis.

A similar analogy may be made for circles and points instead of lines and points. All points on a circle correspond to intersecting circles in the Hough plane. (See [7] for more information.)

The Hough transform has many desirable features. For example, because 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

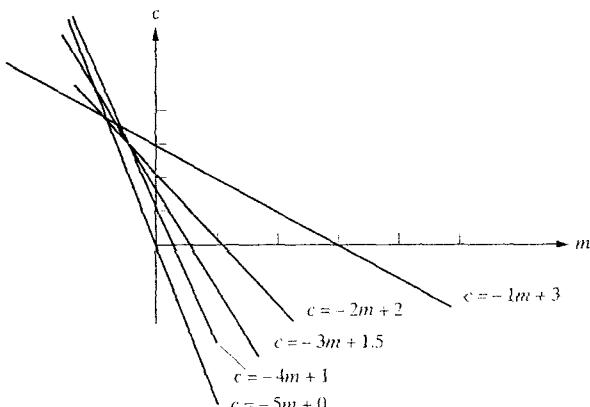


Figure 8.38 Hough transform for Example 8.4.

with each other at the same location, one must define a “circle” (or other boundary) within which, if the lines are intersecting with each other, they are assumed to be intersecting in general. Many variations of the Hough transform have been devised to increase its efficiency and utility for different tasks, including object recognition [13].

8.20 SEGMENTATION

Segmentation is a generic name for a number of different techniques that divide the image into segments of its constituents. The purpose of segmentation 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, by small areas, etc. Each of these entities can then 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 the three-dimensional analysis of objects, models such as cylinders, cones, spheres, and cubes were used as well. Although these shapes and figures do not necessarily match those of any real objects, they provided a means for early developmental work, which evolved into more sophisticated routines and techniques. They also provided a means of developing routines that could process complicated shapes and recognize objects. As an example, with very little processing power the routines could model a tree as a cone mounted on a cylinder and could match the resulting figure with a model of a tree. The tree could thus be expressed with only a few pieces of information, such as the diameters of the cone and cylinder and their heights. Representing *all* the information pertaining to a tree could be staggering in comparison.

8.21 SEGMENTATION BY REGION GROWING AND REGION SPLITTING

Region growing and image splitting are techniques of segmentation, as are edge detection routines. 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 object detection. While an edge detector will find the separation lines of textures, colors, planes, and gray levels, segmentation by regions will naturally result in complete and closed boundaries. (For a survey of other segmentation techniques, see [14].)

Two approaches are used for region segmentation. One is to grow regions by similar attributes, such as gray-level ranges or other similarities, and then try to relate the regions by their average similarities or spatial relationships. The other is region splitting, which will split images into smaller areas by using their finer differences.

One technique of region splitting is *thresholding*. The image is split into closed areas of neighboring pixels by comparing them with thresholding value or range. Any pixel that falls below a threshold (or between a range of values) will belong to

a given region; otherwise, it belongs to another region. Thresholding will split the image into a series of regions or clusters of pixels that have common or similar attributes. Generally, although it is a simple technique, it is not very effective, because choosing an appropriate threshold is difficult. The results are also highly dependent on the threshold value and will change accordingly when the thresholds change. Still, thresholding is a useful technique under certain conditions, such as backlighting, and for images with relatively uniform regions.

In region growing, first nuclei regions are formed on the basis of some specific selection law. (Nuclei regions are the small 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 crystals do in alloys.) The result is a large number of little regions. Successively, these regions are combined into larger regions on the basis of some other attributes or rules. Although these rules will merge many smaller regions together to create a smoother set of regions, they may unnecessarily merge certain features that should not be merged, such as holes, smaller but distinct areas, or distinct areas with similar intensities.

A simple search technique for growing regions for a binary image (or, with the application of thresholding, for gray images as well), uses a bookkeeping approach to find all pixels that belong to the same region [15]. Figure 8.39 shows a binary image. Each pixel is referred to by a pair of index numbers. Assume that a pointer starts at the top and will search for a nucleus to start a region. As soon as a nucleus that does not already belong to another region is found, the program assigns a region number to it. All pixels connected to that nucleus will 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.

It is important to decide what form of connectivity is to be used in growing regions, as the form determines the final outcome. As was discussed in Section 8.16, +4-, $\times 4$ -, H6-, V6-, and 8-connectivity can be used for region growing. In Figure 8.39, the first nucleus is found at cell 2d. Suppose we have chosen +4-connectivity.

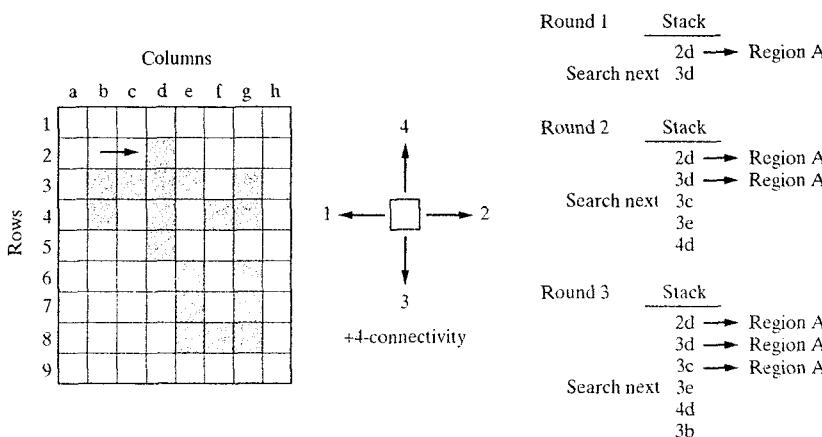


Figure 8.39 Region growing based on a search technique.

Then the program will check the four corresponding pixels around the nucleus to determine their connectivity. If there is an “on” pixel, the index numbers of its location are placed in a stack, the cell is given the region number (A), 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 A region designation, and the process is repeated for the next index number in the stack, 3c. The process continues until the stack is empty.

Notice that, on the basis of +4-connectivity, as the pointer gets to pixels 4f and 6e, it will not assign them to the same region. On the basis of $\times 4$, H6-, V6-, or 8-connectivity, both regions would be part of region A . Otherwise, the pointer continues until new nuclei are found for the next regions — say, region B and region C .

This kind of search technique is nothing more than a bookkeeping instrument to make sure that the computer program can find all connected pixels in the region without missing any. Otherwise, it is a simple search technique.

8.22 BINARY MORPHOLOGY OPERATIONS

Morphology operations refer to a family of operations that are performed on the shape (and thus the 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 its analysis, as well as to reduce the “extra” information that may be present in the image. For example, consider the binary image in Figure 8.40 and the stick figure representing one of the bolts. As we will see later, a moment equation may be used to calculate the orientation of the bolts. However, the same equation can also be applied to 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 sections that follow we will discuss a few of these operations.

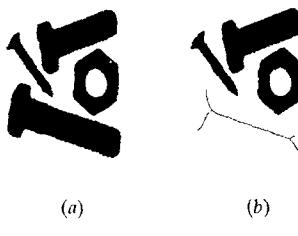


Figure 8.40 The binary image of a bolt and its stick (skeleton) representation.

8.22.1 Thickening Operation

The 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 of Figure 8.40 (a), a thickening operation will reduce the appearance of the threads of the bolts.

This will become important when we try to apply other operations, such as skeletonization, to the object. The initial thickening will prevent the creation of whiskers caused by the threads, as we will see later. Figure 8.41 shows the effect of three rounds of thickening operations on the threads of the bolts.

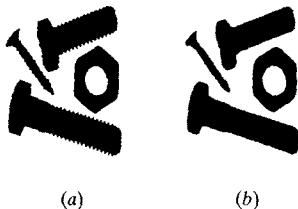


Figure 8.41 The threads of the bolts are removed by a triple application of a thickening operation, resulting in smooth edges.

8.22.2 Dilation

In dilation, the background pixels that are 8-connected to the foreground (object) are changed to foreground pixels. As a result, a layer is effectively added to the object every time the process is implemented. Because dilation is performed on pixels that are 8-connected to the object, repeated dilations can change the shape of the object. Figure 8.42(b) is the result of four dilation operations on the objects in Figure 8.42(a). As can be seen, the objects have bled into one piece. With additional applications of dilation, the objects, as well as the disappearing hole, can become one solid piece and hence cannot be recognized as distinct objects anymore.

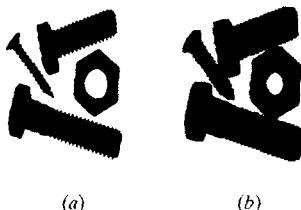


Figure 8.42 Effect of dilation operations. Here, the objects in (a) were subjected to four rounds of dilation (b).

8.22.3 Erosion

In erosion, 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 the operation is performed. Figure 8.43(b) shows the effect of three repetitions of the erosion operation on the binary image in Figure 8.43(a). 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 will remove one pixel from the perimeter (and holes) of the object even if the shape of the object is eventually lost, as in (c) with seven repetitions, where one bolt is completely lost and the nut will soon disappear. The final result of too many erosions will be the loss of the object. That is to say, if the reversing operation of dilation

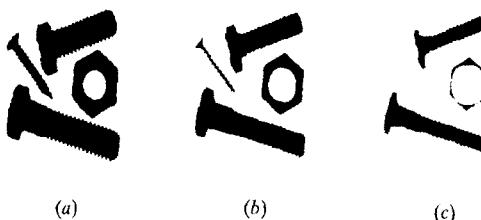


Figure 8.43 Effect of erosion on objects (a) with (b) 3 and (c) 7 repetitions.

tion, which adds one pixel to the perimeter of the object with each pass, is used, 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 a 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 one is interested in identifying the largest object in an image, successive erosions will eliminate all other objects before the largest is eliminated. Thus, the object of interest can be identified.

8.22.4 Skeletonization

A *skeleton* is a stick representative of an object in which all thicknesses have been reduced to one pixel at any location. Skeletonization is a variation of erosion. Whereas in erosion the thickness of an object 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. Also, although in erosion the number of repetitions are chosen by the user, in skeletonization the process automatically continues until all thicknesses are one pixel thick. (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 it—indeed, sometimes much better than the edges. Figure 8.44(b) shows the skeleton of the original objects in Figure 8.44(a). 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 8.45 shows the same objects, thickened to eliminate the threads, resulting in a clean skeleton. Figure 8.45(c) is the result of dilating the skeleton seven times. As can be seen, the dilated objects are not the same

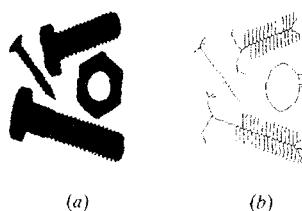


Figure 8.44 The effect of skeletonization on an image without thickening. The threads of the bolts have resulted in the creation of whiskers.

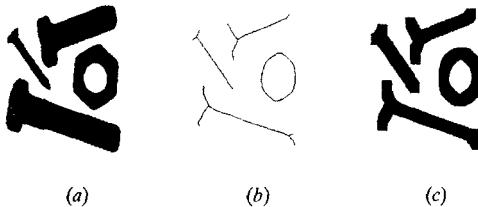


Figure 8.45 The skeleton of the objects in (a) after the application of thickening results in a clean skeleton (b). Part (c) is the dilated image of the skeletons.

as the original ones. Notice how the smaller screw appears to be as big as the bigger bolts.

Although dilating a skeleton will also result in a shape different from that of the original object, skeletons are useful in object recognition, since they are generally a better representation of an object than other representations. When a stick representation of an object is found, it can be compared with the available *a priori* knowledge of the object for matching.

8.22.5 Open Operation

Opening is erosion followed by dilation and causes a limited smoothing of convex parts of the object. Opening can be used as an intermediate operation before skeletonization.

8.22.6 Close Operation

Closing is dilation followed by erosion and causes a limited smoothing of convex parts of the object. Like opening, closing can be used as an intermediate operation before skeletonization.

8.22.7 Fill Operation

The fill operation fills the holes in the foreground (object). In Figure 8.46, the hole in the nut is filled with foreground pixels until it is eliminated.

Information on other operations may be found in vision systems manufacturers' references. Different companies include other operations to make their software unique.

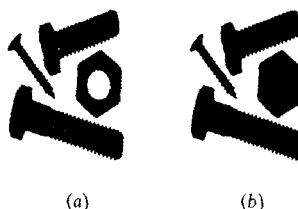


Figure 8.46 As a result of a fill operation, the hole in the nut is filled with foreground pixels and is thus eliminated.

8.23 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 that a gray image is a multilayered three-dimensional image in which the light areas are peaks and the dark areas are valleys. The mask is applied to the image by moving it from pixel to pixel. Wherever the mask matches the gray values in the image, no changes are 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 sections that follow.

8.23.1 Erosion

In this case, each pixel is replaced by the value of the darkest pixel in its 3×3 neighborhood, known as a *min operator* and effectively erodes the object. Of course, the result is dependent on which cells in the mask are 0 or 1. Gray morphology erosion removes light bridges between dark objects.

8.23.2 Dilation

In this case, each pixel is replaced by the value of the lightest pixel in its 3×3 neighborhood, known as a *max operator* and effectively dilates the object. Of course, the result is dependent on which cells in the mask are 0 or 1. Gray morphology dilation removes dark bridges between light objects.

8.24 IMAGE ANALYSIS

Image analysis is a collection of operations and techniques that are used to extract information from images. Among these operations and techniques are object recognition; feature extraction; analysis of the position, size, orientation, and other properties of objects in images; 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 to calculate the position and orientation of an object.

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. Some of these techniques are discussed next.

8.25 OBJECT RECOGNITION BY FEATURES

Objects in an image may be recognized by their features, which may include, but are not limited to, gray-level histograms, morphological features such as area, perime-

ter, number of holes, etc., eccentricity, cord length, and moments. In many cases, the information extracted is compared with a priori information about the object, which may be in a lookup table. For example, suppose that two objects are present in the image, one with two holes and one with one hole. By 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) with information about them in a lookup table, it is possible to determine what each of the two parts are. As another example, suppose that a moment analysis of a known part is performed at different angles, that the moment of the part relative to an axis is calculated for these angles, and that the resulting data are collected in a lookup table. Later, when the moment of the part in the image is calculated relative to the same axis and is compared with the information in the lookup table, the angle of the part in the image can be estimated.

We next discuss a few techniques and different features that may be used for object recognition.

8.25.1 Basic Features Used for Object Identification

The following morphological features may be used for object recognition and identification:

- a. The average, maximum, or minimum gray levels may be used to identify different parts or objects in an image. As an example, assume that the image is divided into three parts, each with a different color or texture that will create different gray levels in the image. If the average, maximum, or minimum gray levels of the objects are found, say, through histogram mapping, the objects can be recognized by comparing them with this information. In other cases, even the presence of one particular gray level may be enough to recognize a part.
- b. The perimeter, area, and diameter of an object, as well as the number of holes it has and other morphological characteristics, may be used to identify the object. The perimeter of an object may be found by first applying an edge detection routine and then counting the number of pixels on the perimeter. The left-right search technique of Section 8.18 can also be used to calculate the perimeter of the object by counting the number of pixels that are on the object's path in an accumulator. The area of the object can be calculated by region-growing techniques. Moment equations can also be used, as will be discussed later. The diameter of a noncircular object is defined as the maximum distance between any two points on any line that crosses the identified area of the object.
- c. An object's aspect ratio is the ratio of the width to the length of a rectangle enclosed about the object, as shown in Figure 8.47. All aspect ratios, except for

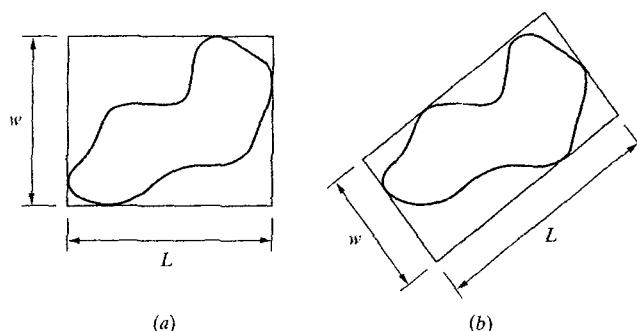


Figure 8.47 Aspect ratio of an object, with minimum aspect ratio shown in part (b).

the minimum aspect ratio, are sensitive to orientation. Thus, the minimum aspect ratio is generally used to identify objects.

- d. Thinness is defined as one of the following two ratios:

$$1. \quad \text{Thinness} = \frac{(\text{perimeter})^2}{\text{area}}. \quad (8.15)$$

$$2. \quad \text{Thinness} = \frac{\text{diameter}}{\text{area}}. \quad (8.16)$$

- e. Moments are discussed in the next section, due to their special importance.

8.25.2 Moments

Imagine an object within a binary image. The object is represented by pixels that are turned on, and the background is represented by pixels that are turned off. This effect can be achieved either by backlighting or by rendering the image in binary form.

Now consider the general moment equation

$$M_{a,b} = \sum_{x,y} x^a y^b, \quad (8.17)$$

where $M_{a,b}$ is the moment of the object within the image with indices a and b and x and y are the coordinates of each pixel that is turned on within the image, raised to powers of a and b , as in Figure 8.48. In such a case, a routine based on Equation (8.17) will first determine whether each pixel belongs to the object (is turned on) and, if so, will then raise the coordinates of the location of the pixel to the given values of a and b . The summation of this operation over the whole image will be the particular moment of the object with a and b values. $M_{0,0}$ is the moment of the object with $a = 0$ and $b = 0$. This means that all x and y values are raised to a power of 0. $M_{0,2}$ means that all x values are raised to the power of 0, all y values are raised to power of 2, etc. All combinations of values between 0 and 3 are common.

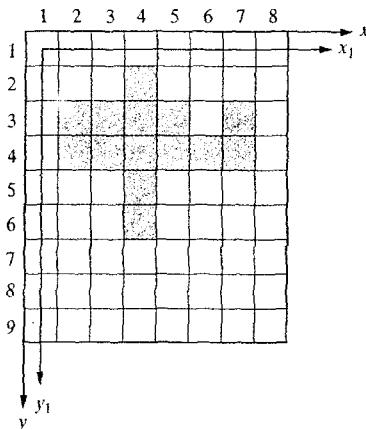


Figure 8.48 Calculation of the moment of an image. For each pixel that belongs to the object, the coordinates of the pixel are raised to the powers indicated by the moment's indices. The summation of the values thus calculated will be the particular moment of the image.

Distances x and y are measured either from a fictitious coordinate frame located at the edge of the image (x,y) or are measured from a coordinate frame 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 coordinate frame is more logical. Note, however, 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 in the second row is $I_{2,4}$. The x distance of the pixel from the x_1y_1 coordinate frame will be 3, whereas the same distance from the xy coordinate is 4 (or, more accurately, 3.5). As long as the same distances are used consistently, the choice is not important.

Based on the preceding discussion, since all numbers raised to the power of 0 are equal to 1, then all x^0 's and y^0 's are equal to 1. As a result, the moment $M_{0,0}$ is the summation of as many 1's as there are "on"-pixels, yielding the total number of "on"-pixels, which is the area of the object. In other words, the moment $M_{0,0}$ is the same as the area of the object. This moment can be used to determine the nature of an object and to distinguish it from other objects that have a different area. Obviously, the moment $M_{0,0}$ can also be used to calculate the area of an object within an image.

Similarly, $M_{0,1}$ is $\sum x^0 y^1$, or the summation of $1 \times y$'s, which is the same as the summation of each pixel area multiplied by its distance from the x -axis. This is similar to the first moment of the area relative to the x -axis. Then the location of the center of the area relative to the x -axis can be calculated by

$$\bar{y} = \frac{\sum y}{\text{area}} = \frac{M_{0,1}}{M_{0,0}}. \quad (8.18)$$

So, simply by dividing the two moments, you can 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 will be

$$\bar{x} = \frac{\sum x}{\text{area}} = \frac{M_{1,0}}{M_{0,0}}. \quad (8.19)$$

This way, an object may be located within an image, independently 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.

Now consider $M_{0,2}$ and $M_{2,0}$. $M_{0,2}$ is $\Sigma x^2 y^2$ 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 8.48 will vary significantly as the object rotates about its center. Suppose that one would calculate the moments of the area about, say, the x -axis, at different orientations. Since each orientation creates a unique value, a lookup table that contains these values can later be used to identify the orientation of the object. Thus, if a lookup table containing the values of the moments of inertia of the known object at different orientations is prepared, the subsequent orientation of the object can be estimated by comparing its second moment with the values in the table. Of course, if the object translates within an image, its moments of inertia will also change. However, if the coordinates of the center of the area of the object are known, then, with a simple application of the parallel axes theorem, the second moments about the center of the area can be calculated independently of their location. As a result, with the use of the moment equations, an object, its location, and its orientation can be identified. In addition to identifying the part, the information can be used in conjunction with a robot controller to direct the robot to pick up the part or operate on it.

Other moments can be used similarly. For example, $M_{1,1}$ represents the product of inertia of the area and can also be used to identify an object. Higher order moments such as $M_{0,3}, M_{3,0}, M_{1,2}$, etc., can also be used to identify objects and their orientations. Imagine two objects that are relatively similar in shape, as in Figure 8.49(a). It is possible that the second moments, areas, perimeters, or other morphological characteristics of the objects may be similar or close to each other, such that they may not be useful in identifying the object. In this case, a small difference between the two objects may be exaggerated through higher order moments, making identification of the object possible. The same is true for an object with a small asymmetry (Figure 8.49(b)). The orientation of the object may be found by higher order moments.

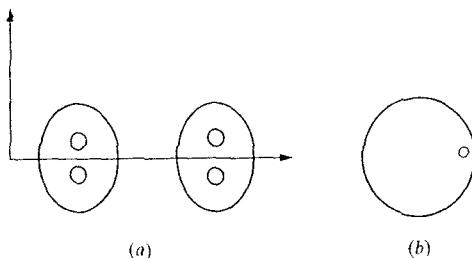


Figure 8.49 Small differences between objects or a small asymmetry in an object may be detected by means of higher order moments.

A *moment invariant* is a measure of an object based on its different moments and is independent of the location and orientation of the object, as well as of the scale factor used to represent the object. There are seven different moment invariants, of which one is

$$MI_1 = \frac{M_{0,0}M_{2,0} - M_{1,0}^2 + M_{0,0}M_{0,2} - M_{0,1}^2}{M_{0,0}^3}. \quad (8.20)$$

(See [6] for the other six moment invariants.)

Example 8.5

For the simple object in the low-resolution image of Figure 8.50, calculate the area, center of the area, and second moments of inertia of the object relative to the x_1 and y_1 -axes.

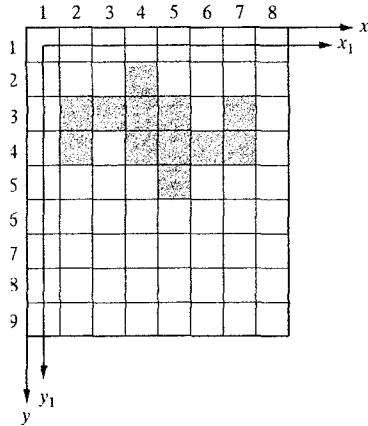


Figure 8.50 Image used for Example 8.5.

Solution Measuring the distances of each pixel from the x_1 - and y_1 -axes and substituting the measurements into the moment equations yields the following results:

$$M_{0,0} = \sum x^0 y^0 = 12(1) = 12;$$

$$M_{1,0} = \sum x^1 y^0 = \sum x = 2(1) + 1(2) + 3(3) + 3(4) + 1(5) + 2(6) = 42;$$

$$M_{0,1} = \sum x^0 y^1 = \sum y = 1(1) + 5(2) + 5(3) + 1(4) = 30;$$

$$\bar{x} = \frac{M_{1,0}}{M_{0,0}} = \frac{42}{12} = 3.5 \quad \text{and} \quad \bar{y} = \frac{M_{0,1}}{M_{0,0}} = \frac{30}{12} = 2.5;$$

$$M_{2,0} = \sum x^2 y^0 = \sum x^2 = 2(1)^2 + 1(2)^2 + 3(3)^2 + 3(4)^2 + 1(5)^2 + 2(6)^2 = 178;$$

$$M_{0,2} = \sum x^0 y^2 = \sum y^2 = 1(1)^2 + 5(2)^2 + 5(3)^2 + 1(4)^2 = 82.$$

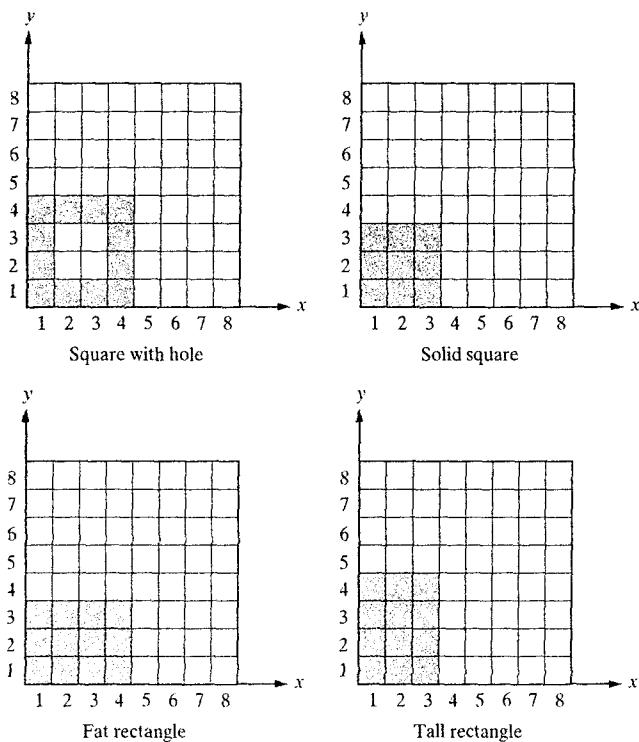


Figure 8.51 Image used for Example 8.6.

The same procedure may be used for an image with much higher resolution; there will just be many more pixels to deal with. However, a computer program can handle as many pixels as necessary without difficulty.

Example 8.6

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 hollows, while the rectangles are 3×4 solids. Through guides, jigs, and brackets, we can be certain that the objects are always parallel to the reference axes, as shown in Figure 8.51, and that the lower left corner of the objects is always at the pixel 1,1. We want to use the moment equations only 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. Use the absolute coordinates of each pixel for distances from the corresponding axes.

Solution Using the moment equations, we calculate the different moments for all four until we find one set that is unique for each object:

Square with hole	Solid square	Fat rectangle	Tall rectangle
$M_{0,0} = 12$	$M_{0,0} = 9$	$M_{0,0} = 12$	$M_{0,0} = 12$
$M_{0,1} = 30$	$M_{0,1} = 18$	$M_{0,1} = 24$	$M_{0,1} = 30$
$M_{1,0} = 30$	$M_{1,0} = 18$	$M_{1,0} = 30$	$M_{1,0} = 24$
$M_{1,1} = 75$	$M_{1,1} = 36$	$M_{1,1} = 60$	$M_{1,1} = 60$
$M_{0,2} = 94$	$M_{0,2} = 42$	$M_{0,2} = 56$	$M_{0,2} = 90$

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.

Example 8.7

For the image of the screw in Figure 8.52, calculate the area, \bar{x} , \bar{y} , $M_{0,2}$, $M_{2,0}$, $M_{1,1}$, $M_{2,0}$ @ \bar{x} , $M_{0,2}$ @ \bar{y} , and the moment invariant. $M_{0,2}$ @ \bar{y} means $M_{0,2}$ about centroidal \bar{y} -axis. Similarly, $M_{2,0}$ @ \bar{x} is the moment of inertia about centroidal \bar{x} -axis.



Figure 8.52 Image used for Example 8.7.

Solution A macro called moments.macro was written for the Optimas™ 6.2 vision software 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 rotations. Every time a part of an image is rotated, since every point in it must be converted with a sine or cosine function, the image changes slightly. Otherwise, the results are consistent. For example, as the part is rotated in place, the location of its center of area does not change. Also, the moment invariant is constant, and using information about the moment of inertia about the centroid, we can estimate the orientation of the part. This information can now be used to identify the object or to direct a robot controller to send the robot arm, with the proper orientation, to the location to pick up the part.

	Horizontal	30°	45°	60°	Vertical
<i>Area</i>	3713	3747	3772	3724	3713
\bar{x}	127	123	121	118	113
\bar{y}	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
<i>Moment Invariant</i>	7.48	7.5	7.4	7.3	7.48
$M_{2,0}$ @ \bar{x}	7.5 E6	5.7 E6	3.94 E6	2.07 E6	0.264 E6
$M_{0,2}$ @ \bar{y}	0.264 E6	2.09 E6	3.77 E6	5.7 E6	7.5 E6

Although the moments.macro program cannot directly be used with other software, we list it next to show how simply a program can be developed to do similar moment op-

erations. The Excel part of the program is nothing more than a simple set of Excel equations that operate on the coordinates of all pixels and, later, are summed up. The following is a listing of the program:

```
/*MOMENTS.MAC PROGRAM Written by Saeed Niku, Copyright 1998
```

This macro checks an active image within the Optimas vision system and records the coordinates of all pixels above the given threshold. It subsequently writes the coordinates into an Excel worksheet, which determines the moments. Moments.mac will then read back and display the data. The DDE commands communicate the data between Excel and the Optimas macro. If the number of coordinates is more than 20,000 pixels, you must change the DDEPoke command below. */

```
BinaryArray = GetPixelRect (ConvertCalibToPixels(ROI));
INTEGER NewArray[]; Real MyArea; Real XBar; Real YBar;
Real Mymoment02; Real Mymoment20; Real Mymoment11; Real VariantM;
Real MyMXBar; Real MyMYBar;

For(XCoordinate = 0; XCoordinate <= (VectorLength(BinaryArray[0])-1); XCoordinate++)
{
    For(YCoordinate = 0; YCoordinate <= (VectorLength(BinaryArray[0])-1); YCoordinate++)
    {
        If (BinaryArray[YCoordinate,XCoordinate] > 100)
        {
            NewArray ::= XCoordinate : YCoordinate;
        }
    }
}
hChanSheet1 = DDEInitiate ("Excel, ""Sheet1");
DDEPoke(hChanSheet1,"R1C1:R20000C2,"NewArray");
DDETernate(hChanSheet1);

Show("Please Enter to Show Values");
hChanSheet1 = DDEInitiate ("Excel, ""Sheet1");
DDERequest(hChanSheet1,"R1C14,"MyArea);
DDERequest(hChanSheet1,"R2C14,"YBar);
DDERequest(hChanSheet1,"R3C14,"XBar);
DDERequest(hChanSheet1,"R4C14,"Mymoment02);
DDERequest(hChanSheet1,"R5C14,"Mymoment20);
DDERequest(hChanSheet1,"R6C14,"Mymoment11);
DDETernate(hChanSheet1);

VariantM=(MyArea*Mymoment20*1000000.0-XBar*XBar
          +MyArea*Mymoment02*1000000.0-YBar*YBar)
          /(MyArea*MyArea*MyArea);

MyMYBar=(Mymoment20*1000000.0-MyArea*XBar*XBar)/1000000.0;
MyMXBar=(Mymoment02*1000000.0-MyArea*YBar*YBar)/1000000.0;

MacroMessage("Area="MyArea,"\\n,""XBar="XBar,"\\n,"
```

```

"YBar=,"YBar,"\\n,""Moment02=,"Mymoment02," x10^6,"
"\\n,""Moment20=,"Mymoment20," x10^6,"\\n,""Moment11="
,Mymoment11," x10^6,"\\n,""Invariant 1="
,VariantM);
MacroMessage("Moment20@Xbar=,"MyMXBar," x10^6,"\\n,"
"Moment02@Ybar=,"MyMYBar," x10^6");

```

8.25.3 Template Matching

Another technique for object recognition is *model*, or *template, matching*. If a suitable line drawing of a scene is found, the topological or structural elements, such as the total number of lines (sides), vertices, and interconnections can be matched to a model. Coordinate transformations (e.g., rotation, translation, and scaling) can be performed to eliminate the differences between the model and the object resulting from differences in position, orientation, or depth between them. This technique is limited by the fact that a priori knowledge of the models is needed for matching. Thus, 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.

8.25.4 Discrete Fourier Descriptors

In a manner similar to a Fourier transform that is calculated for an analog signal, a discrete fourier transform (DFT) of a set of discrete points (such as pixels) can 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 a DFT calculation is a set of frequencies and amplitudes in the frequency domain that describe the spatial relationship of the points in question [16].

To calculate the DFT of a set of points in a plane, assume that the plane is the complex-number plane, such that each point is described by the relationship $x + iy$. If the contour surrounding the set of points 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. These frequencies can then be matched with the frequencies found for possible objects in a lookup table in order 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), and matching 16 frequencies could determine the type of airplane from a large class of planes. An advantage of this technique is that the Fourier transform can be normalized for size, position, and orientation very simply. 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 objects.

8.25.5 Computed Tomography (CT)

Tomography is a technique of determining the distribution of material density in the part being examined. In computed tomography (CT), a three-dimensional

image of the density distribution of the object is reconstructed from a large number of two-dimensional images of the material density taken by different scanning techniques, such as X-rays or ultrasonics. In computed tomography, it is assumed that the part consists of a sequence of overlaying slices. Images of the 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 are stored in a computer and subsequently are reduced to a three-dimensional image of the part's density distribution, which is shown on a CRT. Tomography is the only efficient technique for mapping the internals of objects.

Although this technique is completely different from the other techniques mentioned, it is a viable one 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 differentiate it from other, similar objects. Specifically, in medical situations, a CT scan can be used in conjunction with medical robots, with the three-dimensional mapping of the internal organs of the human body used to direct the robot as it performs surgical operations.

8.26 DEPTH MEASUREMENT WITH VISION SYSTEMS

Depth information is extracted from a scene by means of two basic techniques. One is the use of range finders in conjunction with a vision system and image-processing techniques. In this combination the scenes are analyzed in relation to the information gathered by the range finders about the distances of different portions of an environment or the location of particular objects or sections of the object in that environment. Second is the use of binocular or stereo vision. In this technique, as in humans and animals, two cameras look at a scene simultaneously, or one camera takes an image of a scene, moves a certain distance, takes another image, and continues for multiple images. As long as the scene does not change during this operation, the results will be the same as that obtained with the use of multiple cameras. Since the locations of the two cameras in relation to any particular point in the scene are slightly different, the two cameras will develop slightly different images. By analyzing and measuring the differences between the two scenes, depth information can be extracted.

8.26.1 Scene Analysis vs. Mapping

Scene analysis refers to the analysis of complete scenes of images developed by a camera or another, similar device. In other words, the image is a complete replica of the scene, within the limits of resolution of the device. 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, as well as segmented by edge detection or thresholding. Then the part is isolated by region growing and identified by ex-

tracting its features and comparing them with information in a template or a lookup table. By contrast, *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 are linked with the relative positions of points on the object at discrete locations. Since the image is already sliced, less processing is required in the analysis of mapped images, but less information can be extracted from the scene as well. Each technique has its own merits, benefits, and limitations and is used for different purposes, including navigation.

8.26.2 Range Detection and Depth Analysis

Range measurement and depth analysis are performed using many different techniques, such as active ranging [22], stereo imaging, scene analysis, or specialized lighting. Humans employ a combination of techniques to extract information about the depth and positional relationship between different elements of an image. Even in a two-dimensional image, humans can extract useful information from details such as the changing size of similar elements, vanishing lines, shadows, and the changing intensity of textures and shades. Since many artificial-intelligence techniques are based on, and in fact are studied to gain an understanding of, the way humans do things, a number of depth measurement techniques are designed after similar human operations [17].

8.26.3 Stereo Imaging

An image is the projection of a scene onto the image plane through an ideal lens. Thus, every point in the image will correspond to a certain point in the scene. However, the distance of the point from the plane is lost in this projection and cannot be retrieved simply from the single scene. If two images of the same scene are formed, then the relative depths of different points from the image plane can be extracted by comparing the two images; the differences represent the spatial relationship between different points [18,19]. Humans do the same automatically by combining the two images and forming a three-dimensional one [20,21]. The stereo image used for depth measurement is actually considered to be a 2.5-dimensional image; many more images are required to form a true three-dimensional image.

Depth measurement using stereo images requires two operations:

1. A determination of the point pairs in the two images that correspond to the same point in the scene. This is called the *correspondence* or *disparity* of the point pair. It is a difficult operation to carry out, since some points in one image may not be visible in another, or because sizes and spatial relationships may be different in the two images due to perspective distortion.
2. A determination of the depth or location of the point on the object or in the scene by triangulation or other techniques.

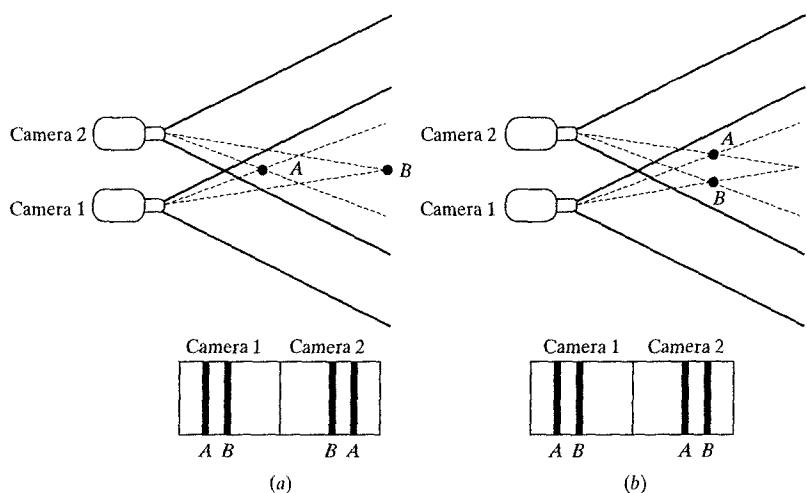


Figure 8.53 Correspondence problem in stereo imaging.

Generally, if the two cameras (or the relative locations of a single camera used twice to get two images of a static scene) are accurately calibrated, triangulation is relatively simple as long as enough corresponding points have been found.

Correspondence points can be determined by matching specific features, such as corners or small segments, from the two images. Depending on their locations, correspondence points can create matching problems. Consider the two marks *A* and *B* in Figure 8.53. In each case, the two cameras will see the marks as shown in (a) and (b). Although the locations of the marks are different, the cameras will see them similarly. As a result, the marks may be located wrongly.

The accuracy of depth measurement in stereo imaging depends on the angle between the two images and thus the disparity between them. 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]. A similar technique was employed in the Stanford Cart, wherein the navigation system would use a camera mounted on a shaft to take multiple images of the scene in order to calculate distances and find obstacles [23].

8.26.4 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 orientation. One of these details 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 shading 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 and difficult.

Another source of information to be used for depth analysis is the texture gradient, or the changes caused in textures as a result of changes in depth. These variations are due to changes in the texture itself, which is assumed to be constant, changes in the depth or distance (scaling gradient), or changes in the orientation of the plane (called the foreshortening gradient). An example is the perceived change in the size of bricks on a wall. By calculating the gradient of the brick sizes on the wall, a depth can be estimated.

8.27 SPECIALIZED LIGHTING

Another possibility for depth measurement is to utilize special lighting techniques that will yield specific results, which can then be used to extract depth information. Most of these techniques have been designed for industrial applications where specialized lighting is possible and the environment is controlled. The theory behind the technique is that if a strip of light is projected over a flat surface, it will generate a straight line in relation to the relative positions and orientations of the plane and light source. However, if the plane is not flat and an observer looks at the light strip in a plane other than the plane of the light, a curved or broken line will be observed. (See Figure 8.54.) By analyzing the reflected light, we can extract information about the shape of the object, its location, and its orientation. In certain systems, two strips of light are used such that in the absence of any object on the table, the two strips will intersect exactly on the surface. When an object is present, the two strips

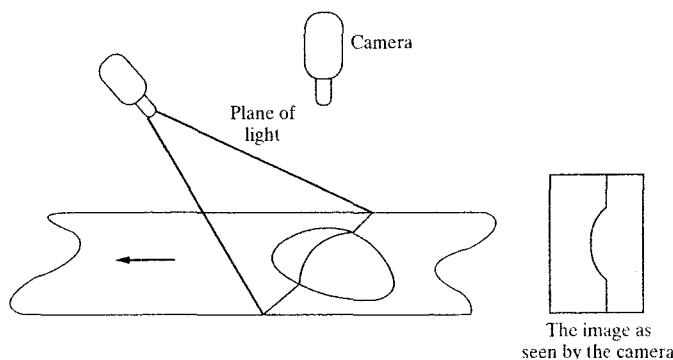


Figure 8.54 Application of a strip of light in depth measurement. A plane of light strikes the object. The camera, located at a different angle than that of the plane of light, will see the reflection of the light plane on the object as a curved line. The curvature of the line is used to calculate depth.

of light develop two reflections. The reflections are picked up by a camera, and a routine calculates the object's features and reports them. A commercial system based on this technique was developed by GM and is called CONSIGHT™. A disadvantage of the technique is that only information about the points that are lit can be extracted. Thus, in order to have information about the complete image, it is necessary to scan the entire object or scene.

8.28 IMAGE DATA COMPRESSION

Electronic images contain large amounts of information and thus require data transmission lines with a large bandwidth capacity. The requirements for the temporal and spatial resolution of an image, the number of images per second, and the number of gray levels (or colors for color images) are determined by the required quality of the images. Recent data transmission and storage techniques have significantly improved image transmission capabilities, including transmission over the Internet.

Although there are many different techniques of data compression, only some of them relate directly 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-frame) methods.

8.28.1 Intraframe Spatial Domain Techniques

Pulse code modulation (PCM) is a popular technique of data transmission in which an analog signal is sampled, usually at the Nyquist rate (a rate that will prevent aliasing), and quantized. The quantizer will have N levels, where N is a power of 2. If N is 8, then 2^8 will yield a quantizer with 256 different gray levels (an eight-bit image quantizer). This arrangement is very common for television-type images and vision systems. Certain other applications (e.g., space and medical applications) use higher resolutions, such as 2^{10} or 2^{12} .

In a technique called *pseudorandom quantization dithering* [24], random noise is added to pixels' gray values to reduce the number of bits necessary to represent the image with the same quality as the original. If the number of bits in a quantizer is reduced without any dithering, contouring will take place. Due to the smaller number of available gray levels in the image, there will be contours of gray levels in the image that make it look like a topological map. (See Section 8.11 and Figure 8.14.) 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. The proper amount of noise will enable the system to have the same resolution while the number of bits is reduced significantly.

Another technique of data compression is to use halftoning. In this technique, a pixel is effectively broken up into a number of pixels by increasing the number of samples per pixel. Instead, every sample is quantized by a simple one-bit binary quantizer into a black or a white pixel. Since the human eye will average the groups of pixels, the image will still look gray and not binary.

Predictive coding refers to a class of techniques that are 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 for many images. Thus, data transmission can be significantly reduced if only the changes between successive images are transmitted.

A predictor is used to predict an optimum value for each pixel, based on the information obtained from the previous images. The *innovation* is the difference between the actual value of the pixel and the predicted value. The predicted value is transmitted by the system to update the previous image. If, in an image, many pixels remain the same, innovations are few and transmission is reduced. In many situations, the images are highly repetitive, elements do not move fast, and large portions of any image, such as the background, do not change. Thus, predictive coding can be used effectively.

In an attempt to reduce the amount of data transmission by the *Voyager 2* spacecraft, its computers were reprogrammed to use a differential coding technique while the vehicle was in space. At the beginning of its journey into space, *Voyager's* system was designed to transmit information about every pixel at a 256-gray-level scale. It took 5,120,000 bits to transmit one image, not including error detection and correction codes, which were about the same length. Beginning with the Uranus flyby, the system was reprogrammed to send only the difference between successive pixels, rather than the absolute brightness of the pixels. Thus, if there were no differences between successive pixels, no information would be transmitted. In scenes such as those in space, where the background is essentially black, many pixels are similar to their neighbors; hence, data transmission was reduced by about 60% [25]. Other examples of fixed background information include TV news sets, theatrical sets, and industrial images.

In *constant-area quantization*, or CAQ [26,27], data transmission is reduced by transmitting fewer pulses at lower resolution in low-contrast areas compared with high-contrast areas. This practice, in effect, takes advantage of the fact that higher contrast areas have higher frequency content and require more information transmission than lower contrast areas.

8.28.2 Interframe Coding Techniques

These methods take advantage of the redundant information that exists between successive images. The difference between interframe coding techniques 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 transmitted.

A simple technique to achieve this aim is to use a frame memory at the receiver. The frame memory will hold an image and will continually show it at the display. When information about any pixel is changed, the corresponding location in the frame

memory is updated. Thus, the rate of transmission is significantly reduced. The disadvantage of this technique is that there is flickering in the presence of rapidly moving elements.

8.29 REAL-TIME IMAGE PROCESSING

In many of the techniques considered so far, the image is digitized and stored before processing. In other situations, although the image is not stored, the processing routines require long computational times before they are finished. This means that, in general, there is a long lapse between the time an image is taken and the time a result is obtained. This may be acceptable in situations in which the decisions do not affect the process. However, in other situations, there is need for real-time processing such that the results are available in real time or in a short enough time to be considered real time. Two different approaches are considered for real-time processing. One is to design dedicated hardware such that the processing is fast enough to occur in real time [28]. The other is to try to increase the efficiency of both the hardware (or parts of it) and the software and thereby reduce processing and computational requirements such that the required times become shorter and closer to real times. In many situations, although a process does not truly happen in real time, compared with the speed of changes in the system and the decision time, the processing time is fast enough to be considered real time.

8.30 HEURISTICS

Heuristics are a collection of rules of thumb that are developed for semiintelligent systems in order to enable them to make decisions based on the current situation. Heuristics are used in conjunction with mobile robots, but have applications in many fields.

Consider a mobile robot that is supposed to navigate through a maze. Imagine that the robot starts at a point and is equipped with a sensor which 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 that the first rule is that, when encountering an obstacle, the robot should turn left. As the robot continues, if it reaches another wall, it will 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, doing so 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 alternative route can be found. As you 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 will be 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 intelligence is not a true intelligence, since the controller is not really making decisions, but merely selects from a set of decisions that have already been made. If a new situation is encountered that is not in the rule base, the controller will not know how to respond [29].

8.31 APPLICATIONS OF VISION SYSTEMS

Vision systems may be used in many different applications, sometimes in conjunction with robotic operations and robots. Vision systems are commonly used for operations that require information from the work environment and that include inspection, navigation, the identification of parts, assembly operations, and communication.

Suppose that in an automatic manufacturing setting, a circuit board is to be manufactured. One important part of this operation is the inspection of the board at different stages — before and after certain operations. A very common application for vision systems is to set up a cell wherein an image of the part to be inspected is taken. Subsequently, image-processing routines are used to modify, improve, and alter the image. Then the processed image is compared with an image from the memory. If there is a match, the part is accepted. Otherwise, the part either is rejected or is repaired. This image-processing-and-analysis operation generally is made up of the processes that were mentioned earlier. Most commercial vision systems have embedded routines that can be called from a macro, making it very easy to set up such a system.

In navigation, a scene is usually analyzed in order to find acceptable pathways, obstacles, and other elements that confront a robot. In some operations, the vision system sends its information to an operator, who controls the motions from a distance. This configuration is very common in telerobotics, as well as in space applications [30]. In some medical applications, too, the surgeon guides the device, be it a surgical robot or a small investigative, exploratory device that produces an angiogram through its operations. 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.

In another application [31,32], an inexpensive laser diode was mounted next to a camera. The projected laser light was captured by the camera and was used to 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 spot and separate it from the rest of the scene, histogram and thresholding operations were used. Subsequently, the circle was identified and then skeletonized until only its center 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, fundamental knowledge about vision systems enables one to proceed with an application and adapt to it what vision systems have to offer.

8.32 DESIGN PROJECT

There are many inexpensive digital cameras on the market that can be used to create a simple vision system. These cameras 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 software to capture and digitize an image. Standard VHS video cameras can also be used in conjunction with products such as Snappy™, which allows you to digitize and save images in your computer. You may also use a standard digital camera to capture an image and to download the image into your computer. In this case, although you can capture an image for later analysis, due to the additional steps involved in downloading the image from the camera to the computer, the image is not available for immediate use.

There are many simple programs, such as Adobe Photoshop™, that have many routines similar to those we have discussed in this chapter. Additional routines may be developed using common computer languages, such as C. The final product will be a simple vision system with some vision capability that can be used to perform vision-related tasks. This development may be done independently or in conjunction with a three-axis robot and can include routines for identifying and picking up parts, developing mobile robots, and creating other, similar devices.

All images shown in this chapter were captured and processed by the vision systems including MVS909™ and Optimas™ 6.2 vision systems, in the Mechanical Engineering Robotics laboratory at Cal Poly, San Luis Obispo, CA. You may also develop your own simple vision system using other programming languages and systems that can handle an image file. Among these systems are Excel™, LabView™, and other development systems.

8.33 SUMMARY

In this chapter, we studied the fundamentals of capturing an image, image processing to modify, alter, improve, or enhance an image, and image analysis through which data can be extracted from an image for subsequent application. Vision systems may be used for a variety of applications, including manufacturing, surveillance, navigation, and robotics. Vision systems are flexible, inexpensive, powerful tools that can be used with ease.

There are countless different routines that can be used for variety of purposes. Most of these 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 concentrated on these techniques, which enable you to adopt, develop, and use other routines and techniques for other applications. The

advances in technology have created tremendous opportunities for vision system and vision analysis. There is no doubt that the trend will continue into the future.

REFERENCES

1. Madonick, N., "Improved CCDs for Industrial Video," *Machine Design*, April 1982, pp. 167-172.
2. Wilson, A., "Solid-State Camera Design and Application," *Machine Design*, April 1984, pp. 38-46.
3. "A 640×486 Long-Wavelength Infrared Camera," *NASA Tech Briefs*, June 1999, pp. 44-47.
4. Meagher, J., *Fourier.xle Program*, Mechanical Engineering Department, California Polytechnic State University, San Luis Obispo, CA, 1999.
5. Doudoumopoulos, Roger, "On-Chip Correction for Defective Pixels in an Image Sensor," *NASA Tech Briefs*, May 2000, p. 34.
6. Gonzalez, R. C., Richard Woods, *Digital Image Processing*, Addison-Wesley, 1992.
7. Low, Adrian, *Introductory Computer Vision and Image Processing*, McGraw-Hill, 1991.
8. Horn, B.K.P., *Robot Vision*, McGraw-Hill, 1986.
9. Hildreth, Ellen, "Edge Detection for Computer Vision System," *Mechanical Engineering*, August 1982, pp. 48-53.
10. Olson, Clark, "Image Smoothing and Edge Detection Guided by Stereoscopy," *NASA Tech Briefs*, September 1999, pp. 68-69.
11. Groover, M. P., et al., *Industrial Robotics, Technology, Programming, and Applications*, McGraw-Hill, 1986, p. 177.
12. Hough, P. V. C., *A Method and Means for Recognizing Complex Patterns*, U.S. Patent 3,069,654, 1962.
13. Illingworth, J., J. Kittler, "A Survey of the Hough Transform," *Computer Vision, Graphics, and Image Processing*, Vol. 44, 1988, pp. 87-116.
14. Kanade, T., "Survey; Region Segmentation: Signal vs. Semantics," *Computer Graphics and Image Processing*, Vol. 13, 1980, pp. 279-297.
15. Snyder, Wesley, *Industrial Robots: Computer Interfacing and Control*, Prentice Hall, 1985.
16. Gonzalez, Rafael, P. Wintz, *Digital Image Processing*, 2d ed., Addison-Wesley, Reading, MA, 1987.
17. 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.
18. Nevatia, R., *Machine Perception*, Prentice-Hall, 1982.
19. Fu, K. S., Gonzalez, R. C., Lee, C. S. G., *Robotics; Control, Sensing, Vision, and Intelligence*, McGraw-Hill, 1987.
20. Marr, D., T. Poggio, "A Computational Theory of Human Stereo Vision," *Proceedings of the Royal Society, London*, B204, 1979, pp. 301-328.
21. Marr, D., *Vision*, Freeman and Co., 1982.
22. Pipitone, Frank, T. G. Marshall, "A Wide-field Scanning Triangulation Rangefinder for

- Machine Vision," *International Journal of Robotics Research*, Vol. 2, No. 1, Spring 1983, pp. 39–49.

23. Moravec, H. P., "Obstacle Avoidance and Navigation in the Real World by Seeing Robot Rover," *Stanford Artificial Intelligence Laboratory Memo*, AIM-340, Sep. 1980.

24. Thompson, J. E., "A 36-Mbit/s Television Coder Employing Psuedorandom Quantization," *IEEE Transactions on Communication Technology*, COM-19, No. 6, December 1971, pp. 872–879.

25. Goldstein, Gina, "Engineering the Ultimate Image, The Voyager 2 Mission," *Mechanical Engineering*, December 1989, pp. 30–36.

26. 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.

27. 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.

28. McHugh, Peter, "Vision and Manufacturing," *NASA Tech Briefs*, June 1999, pp. 36, 37.

29. Chattergy, R., "Some Heuristics for the Navigation of a Robot," *International Journal of Robotics Research*, Vol. 4, No. 1, Spring 1985, pp. 59–66.

30. Ashley, Steven, associate editor, "Roving Other Worlds by Remote," *Mechanical Engineering*, July 1997, pp. 74–76.

31. 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.

32. Niku, S. B., "Camera Calibration and Resetting with Laser Light," *Proceedings of the Third International Conference on Mechatronics and Machine Vision in Practice*, September 1996, Guimarães, Portugal, Vol. 2, pp. 223–226.

PROBLEMS

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 0's and 1's or gray-level numbers in the file:

In a binary image, the 0's represent "off," dark, or background pixels, while 1's represent "on," light, or object pixels. In gray images, each pixel is represented by a corresponding grayness 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 the result of the operation and m and n are the row and column indices, respectively, of the resulting file.

Alternatively, you may use your own graphics system or any commercially available graphics language to create, access, and represent an image.

1. Write a computer program for the application of a 3×3 averaging convolution mask onto a 15×15 image.
2. Write a computer program for the application of a 5×5 averaging convolution mask onto a 15×15 image.
3. Write a computer program for the application of a 3×3 high-pass convolution mask onto a 15×15 image for edge detection.
4. Write a computer program for the application of an $n \times n$ convolution mask onto a $k \times k$ image. Write the routine such that the user can choose the size of the mask and the values of each mask cell individually.
5. Write a computer program that will perform the left-right search routine for a 15×15 image.
6. Using the left-right search technique, find the outer edge of the object shown in Figure P.8.6.

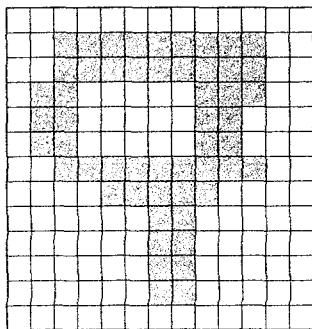


Figure P.8.6

7. 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 another nucleus until all object pixels have been checked.
8. Using +4-connectivity logic and starting from the pixel 1,1, write the sequence of pixels, in the correct order, that will be detected by a region-growing routine for the image shown in Figure P.8.8.

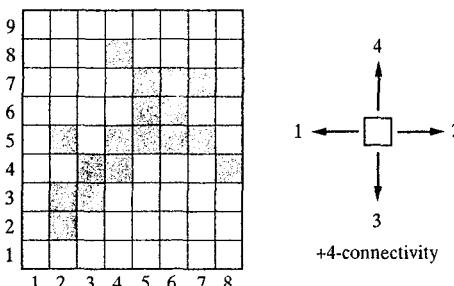


Figure P.8.8.

9. Write a computer program in which different moments of an object in an image can be calculated. The program should query you for moment indices. The results may be reported to you in a new file or may be stored in memory.
10. For the 10×10 binary image of the key shown in Figure P.8.10, calculate the following:
 - Perimeter, based on the left-right 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.

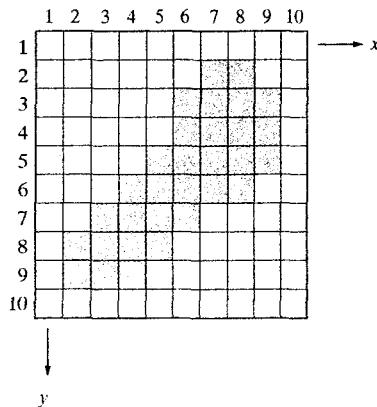


Figure P.8.10.

9

Fuzzy Logic Control

9.1 INTRODUCTION

Consider the following statement: Tuesday, October 26 was supposed to be a very warm day in San Luis Obispo, and in fact it turned out to be pretty hot. When the robotics lab was opened in the morning, we found out that the steam line had leaked into the room, and much heat and humidity had been released into the environment. When the hydraulic power unit for the robots was turned on, it added even more heat to the lab, raising the temperature even further. Eventually, it got so hot that we had to bring in large fans to cool down the lab a bit to make it a little more comfortable for students.

This true statement is a very good example of what fuzzy logic is about. Let's look at the statement again, noticing the italicized words:

Tuesday, October 26 was supposed to be a *very warm* day in San Luis Obispo, and in fact it turned out to be *pretty hot*. When the robotics lab was opened in the morning, we found out that the steam line had leaked into the room, and *much* heat and humidity had been released into the environment. When the hydraulic power unit for the robots was turned on, it added even more heat to the lab, raising the temperature even *further*. Eventually, it got *so hot* that we had to bring in *large* fans to cool down the lab *a bit* to make it *a little more* comfortable for students.

As you see, in this statement, there are a number of "descriptors" used to state certain conditions that are not very clear. For example, when we state that the day was supposed to be very warm, what do you think it was supposed to be? 85°F? Or maybe 100°F? In fact, if you live in San Luis Obispo, even 80°F may be a warm day. Then, as you can see, this description of the temperature is in fact fuzzy. It is not very clear what the temperature may be. As you go on, the statement continues to be fuzzy. We also don't know exactly what is meant by so hot, or a bit, or large fans. How large? How much cooler did the temperature get when we turned on the fans? Now I suggest that you read the paragraph you are reading once again and see how

many other fuzzy descriptions are used in addition to the ones we were discussing. So, obviously, the fuzzy statements are very common in everyday speech, and they are consistently used in all matter of conversation.

What makes this even more interesting is that these fuzzy descriptions of event and other phenomena are context dependent. A warm day in Alaska means something very different from one in Dallas or Rio de Janeiro, and warm day in Dallas is very different if it happens in the summer than in winter. It is expected that summer days in Dallas are actually hot. So, an 85°F temperature will feel relatively cool, whereas the same temperature in the middle of January in Dallas will feel much warmer. Consider the example of the description of pain by a patient to a doctor. If a young child has just fallen and says "it really hurts," the understanding of the doctor will be different than someone who has had a serious accident and says "it really hurts." The description of intensity of pain (as well as its understanding by someone else) while delivering a baby will also be different than the pain during the stitching of a wound. So, we must extend the fuzziness of descriptions to the context as well.

We have learned to listen to these fuzzy descriptions and understand the meaning of them, even if not very accurately and even if it is related to the context and the person who is describing it and the conditions under which it is described. We have learned to associate a certain range of meanings to particular descriptions, as they relate to other conditions, and we are capable of making inferences related to the fuzzy descriptions. In everyday conversations, this seems to work, even if we have to describe ourselves later, or if we have to ask more questions to clarify the meanings. However, let's consider three simple examples where we need to have a better way of describing the situation.

First, let's assume that there is an "expert machine" that is used to prescribe medicine to patients in remote areas where there is no access to doctors for routine problems and that having a doctor visit a patient on the phone is not practical. The system will have to ask the patient about his or her condition, symptoms, pain, fever, coughing, etc., and then it will have to compare these conditions to a lookup table or bank of possible reasons, from which the doctor may diagnose the cause and prescribe a medicine for it. Now suppose that a patient is describing a sore throat. If the sore throat is "really bad," or if the fever is "about" 100°, what is the expert system to understand?

Second, consider a temperature-control thermostat. Suppose that we want to control the temperature at a baseline of 75°, so that if the temperature is more than the baseline, the air conditioning should turn on, and if it is cooler than the baseline, it should turn off. To do this, we want to set the thermostat control to 75°. In a simple microprocessor control, we may have a control statement such as

IF TEMPERATURE \geq 75° TURN ON A/C. (9.1)

This means that as soon as the temperature is 75° or more, the air conditioning will turn on. However, as you notice, the system will not turn on if temperature is 74.9°. Consider the fuzzy descriptions we saw earlier; is this what we intend?

Third, consider a washing machine. In most machines, except for a simple time of wash, there are no other choices that the user can make based on how much is

being washed, and how dirty the clothes are. Would it not be better if there would be a system where the water is tested for its cleanliness, and the washing (scrubbing) mode is adjusted accordingly? In this case, we would 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 to be dirty (and how dirty)?

9.2 FUZZY CONTROL: WHAT IS NEEDED?

Let's reconsider Equation (9.1):

IF TEMPERATURE $\geq 75^\circ$ TURN ON A/C.

One way to improve the flexibility of this control statement is to add another statement to it that would turn on the air conditioning at a little lower temperature, but at slightly lower power setting (assuming that it is possible to change the power setting of the air conditioning system). Then we may change the preceding control statement and add another control statement to it as follows:

IF TEMPERATURE $\geq 70^\circ$ TURN ON A/C 90% FULL POWER, (9.2)

IF TEMPERATURE $\geq 75^\circ$ TURN ON A/C FULL POWER. (9.3)

Now we have added a bit of flexibility to the system, as it is not just dependent on one single value to operate, but it will also react differently depending on the temperature. Notice that still for each statement, the controller reacts to one single value. As a result, even at 74.99° the system is at 90% full-power.

There are still two major problems with this type of control statement. One is that if we intend to have control over a very large range of values, hundreds of this type of statements will be necessary to cover small variations to desired values. Imagine that we desire to have control over every 0.1°F variation in temperature in a chemical process, which may vary $\pm 10^\circ\text{F}$. There will be about 200 control statements! Second, even if we do this and write control statements for all possible variations in a variable, we still cannot relate the statements to everyday spoken words. As a result, the medical expert system of the preceding example will not be able to communicate with the patients, and the washing machine will not be able to relate to dirty and clean water. How about barbecuing?

This is why we need to find a way of systematically defining spoken fuzzy descriptors into useful engineering descriptions that a system can use. This is done using a technique called fuzzy inference control. In the next few 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 base) can be written, and how to convert the results into useful engineering output (called defuzzification). The fuzzy control idea started with the publication of a paper by Lotfi Zadeh [1], and since then, much has been added to the field. Although much more can be discussed about fuzzy logic, we will only discuss some fundamentals of fuzzy logic in this book as it relates to developing a fuzzy logic controller for a device and as it relates to simple machines, including a robot. For more information, please refer to [2,3,4].

9.3 CRISP VALUES VS. FUZZY VALUES

In the aforementioned examples, all values that were mentioned in the statements are called “crisp” values. A crisp value is clearly defined value with one interpretation. A crisp value of 75°F 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 many different meanings may be interpreted from it, or different values may be associated with it.

9.4 FUZZY SETS: DEGREES OF MEMBERSHIP AND TRUTH

To be able to use a fuzzy value in a control setting, we define a fuzzy set, in which a set of values all related to a fuzzy value are members of this fuzzy set. Each value in the fuzzy set has a degree of membership within the set, varying from 100% membership (1) to 0% membership (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 in it where each value has a degree of truth, varying from 100% true to 0% true.

To understand this, let's once again consider the washing machine. If we define clean water to be purely clean water, then as a crisp value, when the water is purely clean (no other material in the water, measured by any method) then the clean-water statement is true, and in all other cases, even if there is a slight amount of dirt in it, the statement is false; the water is not clean. No deviation is allowed in that definition. However, in a fuzzy set defining clean water, a purely clean water will have a degree of membership of 100% (or 1) in the set; it is purely clean water. However, water with slight amount of dirt is still somewhat clean, if not 100%, perhaps 95%. Water with a little more dirt in it is not even 95% clean, but perhaps 90% clean. So, every value in the set relates to some definition of clean water, with only one single crisp value in it (called a singleton), but also containing countless other clean-water possibilities with different degrees of membership and different levels of truth attached to them. In this case, dirty water could still be a part of a clean-water set, but may have a degree of membership of 0% (or 0) with 0 degree of truth. On the other hand, if we also define a fuzzy set called dirty water, the purely clean water mentioned before will have a degree of membership (and truth) of zero in the dirty-water set, while the dirty water will have a 100% degree of membership (and truth) in that set. The water with 90% degree of membership in clean water may have a 15% degree of membership in the dirty-water set as well. So if we have defined two sets, the water under consideration may have two defined values, one in each set, each with a different degree of membership.

Considering the following general crisp rule:

$$\text{IF RULE THEN CONSEQUENCE.} \quad (9.4)$$

If RULE is 100% true, CONSEQUENCE will be executed.

However, with a fuzzy rule, values are not necessarily 100% true, but have degrees of membership in the set. The corresponding defined membership value is used in the rule 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 few sections, we will discuss the process of fuzzification, development of rule bases, and defuzzification.

9.5 FUZZIFICATION

Fuzzification is the process of converting input and output values into their membership functions. The result of fuzzification is a set of graphs that describe the degree of membership of different values in different fuzzy variables.

For example, consider temperatures. To be able to define temperature variables in a fuzzy form, we will divide the desired range of temperatures into a number of sets. For the purpose of illustration, let's use sets of Very-hot, Hot, Warm, and Cold. Of course, the range may be divided into more or fewer sets. Next, we will assign values to each set. Since most natural phenomena follow the bell shaped curves, one should expect to assign values to the set according to a bell shaped Natural curve, as shown in Figure 9.1(a). However, it is actually relatively difficult to work with the Natural curve, while approximating the curve to a simple triangle, as in Figure 9.1(b), will make it much easier to work with, even if not as accurate.

Then, to create the fuzzy sets for the temperature values, say, between 60° and 100°F, we will assign corresponding temperature ranges to the above definitions, as shown in Figure 9.2. Each set contains a set of temperatures, where every temperature has a degree of membership as shown. As was mentioned before, any temperature, say, 78° would have corresponding membership values in different sets. In this

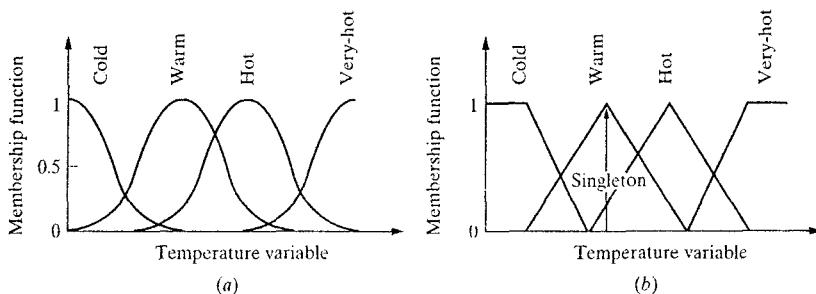


Figure 9.1 Fuzzification of crisp values.

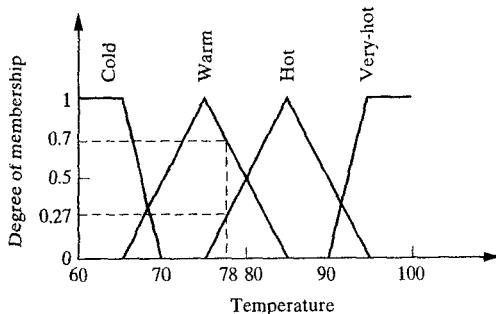


Figure 9.2 Fuzzy sets for a temperature variable.

case, the values are 0.27 in Hot and 0.7 in Warm. Two temperature sets Cold and Very-hot are not triangular shape, since they have either a lower limit or a higher limit, and are unlimited on the opposite side.

The membership functions modeled in this manner are easy to formulate. To define each part of the function, two points on each straight line is expressed. All points on the line can then be easily identified. For example, the membership function for Very-hot and Hot can be expressed as:

$$\text{Very-hot: } @90,0,@95,1,@100,1, \quad (9.5)$$

$$\text{Hot: } @75,0,@85,1,@95,0. \quad (9.6)$$

Based on these definitions, all membership values on all sets can be calculated from the limits shown.

9.6 FUZZY INFERENCE RULE BASE

Fuzzy inference rule base is the controller part of the system, and is based on truth table logic. The rule base is a collection of rules related to the fuzzy sets, the input variables, and the output variables and is meant to allow the system to decide what to do in each case. It generally takes 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>
```

For example, for a system where the temperature is one input variable, humidity is the second input variable, and power setting of the air conditioning system is the output variable, a fuzzy rule may be

IF temperature is Hot and humidity is Humid then power is High. (9.7)

However, the logic of the same rule may be written as

IF temperature is Hot or humidity is Humid then power is High.

Obviously, these two rules will behave differently. For commonly used truth tables, in the first case, both conditions must be true for the consequence, while in the second case, either condition will result in a consequence. However, remembering that these are all fuzzy values and not crisp values and that all variables have membership values in the sets, they do not result in true or false values. Thus, to evaluate the “and” and the “or” rules, we need to define what they mean. In fuzzy logic rule base evaluation between two values, the following will be used:

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, a logic 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. The engines (of course as part of a whole system) can be written by the user, or alternatively, commercial engines such as FIDE™ [5], Fudge™ [6], and Fuzzy Knowledge Builder™ [7] can be bought from vendors.

The total number of rules in a rule base 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 in them, the total number of rules will be equal to $R = m \times n \times p$.

Equation (9.7) can also be demonstrated graphically in order to assist the designer in visualizing the relationships. Figure 9.3 is the graphical representation of this equation. When all the rules are determined, they all may be represented together in a similar manner.

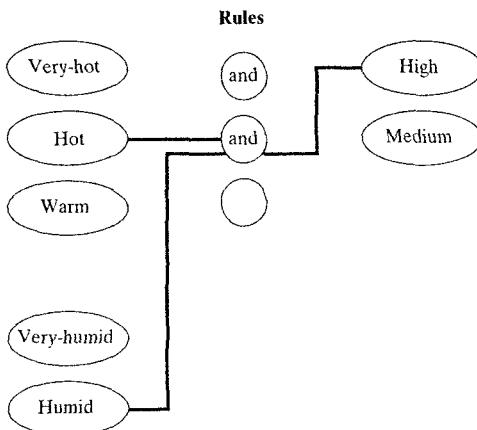


Figure 9.3 Graphical representation of rules.

9.7 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. For example, suppose that the output power setting for an air-conditioning system is fuzzified into OFF, LOW, MEDIUM, and HIGH. The result of rule base evaluation 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 which can be sent to the air conditioning control system.

There are a number of different possibilities for defuzzification. We will consider two common and useful techniques called center of gravity and Mamdani's inference method [8].

9.7.1 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. When these equivalent values for each set are added together and then normalized by summing the output membership values, an equivalent output value is defined for the output. 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 the preceding together and divide by the summation of output membership degrees.

For example, suppose that the values obtained for the output of the air-conditioning system membership sets are 0.4 for LOW and 0.6 for 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 air conditioning would then be

$$\text{Output} = \frac{0.4 \times 30\% + 0.6 \times 50\%}{0.4 + 0.6} = 42\%.$$

9.7.2 Mamdani's Inference Method

In this method, the membership function of each set is truncated at the corresponding membership value, as in Figure 9.4. The resulting membership functions are then added together as an "or" function. This means that all areas that are repeated are superimposed over each other as one layer only. The result will be a new area that is a representative of all areas, once each. The center of gravity of the resulting area will be the equivalent output. Mamdani's method can be summarized as follows:

- (1) Truncate each output membership function at its corresponding membership value, which is found from the rule base.
- (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.

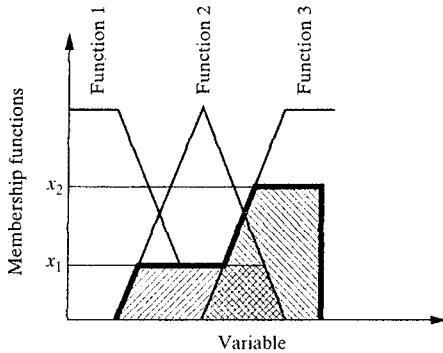


Figure 9.4 Defuzzification based on Mamdani's method.

Most programs calculate this area by integrating the area under the curve. Hand calculations can be done with any technique.

Through this process of fuzzification, application of rules base, and defuzzification, an output value is calculated that can be applied to the output. The following examples demonstrate this procedure for the calculation of an output value:

Example 9.1

A cooling system is to be controlled by fuzzy logic. The two inputs are temperature and humidity, and the output is the power setting of the air conditioning system. Design the fuzzy logic control system.

Solution Based on the foregoing discussion, we will follow the three steps to design the system:

1. Fuzzification: In this part, we develop the fuzzy sets relating to the two inputs and the output. We assume that the range of temperatures we are interested in is 60 to 100° F, the desired range of humidity is 0 to 100%, and the power setting can be 0 to 100%. Figure 9.5 demonstrates the three fuzzy sets for the two inputs and one output. The temperature range is divided into four membership functions of Cold, Warm, Hot, and Very-hot. Temperatures below 60° F are all considered to be part of Cold, and temperatures above 100° F are all part of the Very-hot set. Similarly, the range of humidity is divided into three membership functions of Very-humid, Humid, and Dry. All humidity values below 40% is considered to be Dry. The output power setting is also divided into four membership functions of Fast, Medium, Slow and Off.

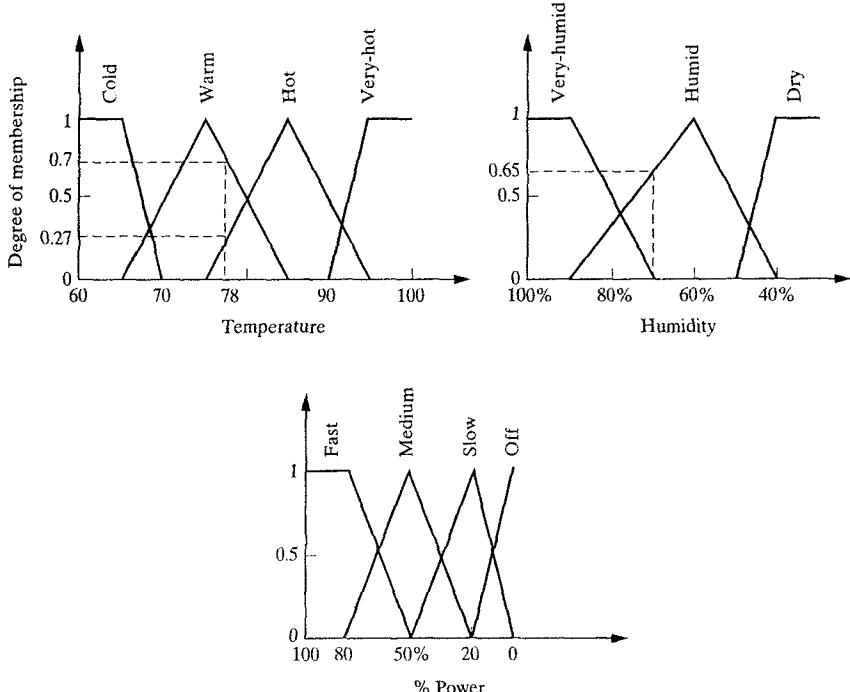


Figure 9.5 Fuzzification of inputs and outputs for Example 9.1.

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, as will be seen later, the response of the system will be studied later, and if necessary, adjustments will be made.

2. Development of the Rules Base. Since there are four membership functions for temperature and three membership functions for humidity, there will be a total of $4 \times 3 = 12$ rules. We will demonstrate the 12 rules both graphically (Figure 9.6) and symbolically. Notice that in each rule (or control law), one membership function for every input is considered. The consequence for each rule is then picked based on the experience of the designer and the necessities of the design. For example, in Rule 1, if temperature is Very-hot and humidity is Very-humid, the consequence, based on experience and the desired result of the rule would be to have the system work at 100% power setting, or Fast. However, in Rule 10, the consequence for Cold and Very-humid could either have been Slow or Off. Which one is correct? We will not really know that

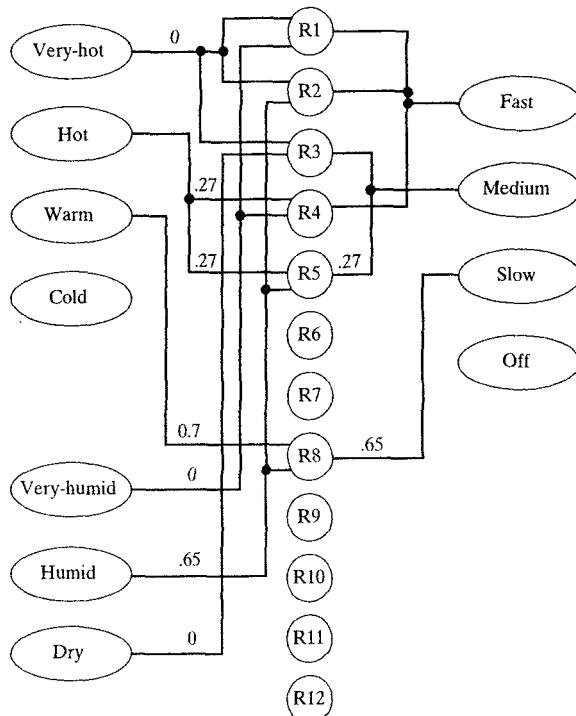


Figure 9.6 Graphical representation of some of the rules in the Rules Base “for” Example 9.1. Other rules can be similarly demonstrated, but are not shown for clarity.

- Rule 1: If temperature = Very-hot and humidity = Very-humid then power = Fast.
- Rule 2: If temperature = Very-hot and humidity = Humid then power = Fast.
- Rule 3: If temperature = Very-hot and humidity = Dry then power = Medium.
- Rule 4: If temperature = Hot and humidity = Very-humid then power = Fast.
- Rule 5: If temperature = Hot and humidity = Humid then power = Medium.
- Rule 6: If temperature = Hot and humidity = Dry then power = Medium.
- Rule 7: If temperature = Warm and humidity = Very-humid then power = Medium.
- Rule 8: If temperature = Warm and humidity = Humid then power = Slow.
- Rule 9: If temperature = Warm and humidity = Dry then power = Slow.
- Rule 10: If temperature = Cold and humidity = Very-humid then power = Slow.
- Rule 11: If temperature = Cold and humidity = Humid then power = Off.
- Rule 12: If temperature = Cold and humidity = Dry then power = Off.

until the output of the rule base is simulated and the result is analyzed. If the output is not as desired, the rules (or membership functions) will be adjusted until a satisfactory result is obtained. Please also notice that some of the rules could have been based on “and,” while others were based on “or.”

To understand how the results are found, let's look at some numbers. Suppose that the present temperature is 78° F and humidity is 70%. As shown in Figure 9.5, the resulting membership values will be 0.7 Warm, 0.27 Hot, and 0.65 Humid. All other membership values are zero.

Substituting these values into the corresponding rules (also shown graphically in Figure 9.6) yields output membership values of 0.27 Medium and 0.65 Slow. Remember that since we are using “and” logic, the minimum value between two numbers are chosen. So, for example, in Rule 5, the smaller of 0.65 and 0.27 is chosen.

3. Defuzzification: Now that we have found the output membership values, we have to defuzzify the values to get a crisp power setting for the system. We will calculate the output value based on both center-of-gravity method and Mamdani's inference method.

For the center-of-gravity method, we will multiply the output membership values by their corresponding singleton values and then divide by the sum of membership values to get

$$\text{Power} = \frac{0.65 \times 20\% + 0.27 \times 50\%}{0.65 + 0.27} = 29\%.$$

For Mamdani's inference method, we will first truncate the Medium and Slow functions at 0.27 and 0.65 values, combine the two into a single area, and then calculate the center of gravity of the resulted area, as shown in Figure 9.7:

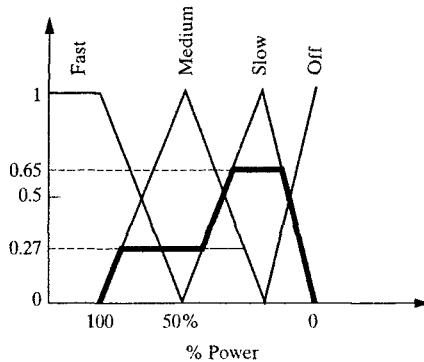


Figure 9.7 Application of Mamdani's inference method.

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 at 34% power rating, which is somewhat different from the center-of-gravity method.

9.8 SIMULATION OF FUZZY LOGIC CONTROLLER

It is necessary that the rules developed for the fuzzy system be simulated to ensure that they yield acceptable results for all input values. This is usually done through fuzzy logic programs, such as FIDE, FUDGE, and other programs mentioned earli-

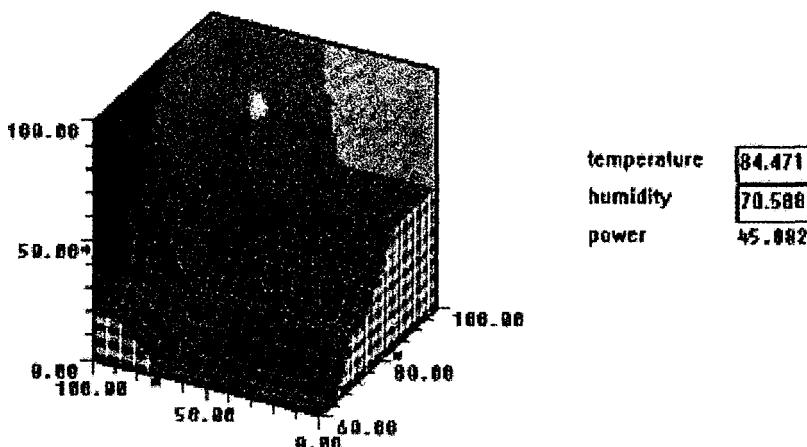


Figure 9.8 The three-dimensional output result of the air-conditioning example generated by the FIDE program. The graph can be used for modifying and adjusting the fuzzy sets or the rules for best output result.

er. In such programs, the membership functions are entered, and the rules are programmed. The program simulates the fuzzy control system by running the fuzzy inference engine and calculating the output for all possible input values (with a chosen step size) and plotting the output values. 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 rule base for the fuzzy sets is modified until the output curves are as desired. Figure 9.8 is the three-dimensional depiction of the output of the air conditioning system from FIDE program. 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 will then run the machine or the system based on the fuzzy program. Although the process seems to be long, it actually is relatively easy to do, and it does add interesting “intelligence” to a machine.

9.9 APPLICATIONS OF FUZZY LOGIC IN ROBOTICS

Fuzzy logic control systems can be used for both controlling robots, as well as for adding intelligence to applications where other systems may be inadequate or difficult to use. For example, in one application, fuzzy logic was used to directly control the torque output of a switched reluctance motor by a current modulation scheme [9]. 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 perhaps be more appropriate, if not the only way to control a function. It is for this reason that the discussion on fuzzy logic has been presented here.

Through these applications, a robot can become unique, more intelligent, or more useful. For 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, robot's power, etc. Or imagine a robot whose end effector must exert a force proportional to two other inputs, say, the size of a part and its weight. 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. In addition, many peripheral devices are added to robots or work with a robot through their own controller. In these cases, a separate microprocessor may be used to control the function of the device independent of the robot controller. When appropriate, fuzzy logic may be incorporated into the microprocessor for its own functions.

Example 9.2

Design a fuzzy logic system for the motion control of a mobile robot on rough terrain.

Solution We will assume that the inputs to the system are the slope of the terrain and the terrain type, while the output is the robot's speed. We will assume that the slope can range between -45 and $+45$ degrees, divided into Large-Negative, Negative, Level, Positive, and Large-Positive. We will further assume that the terrain can be Very-Rough, Rough, Moderate, and Smooth. The output speed can range between 0 and 20 miles per hour and is divided into Very-Slow, Slow, Medium, Fast, and Very-Fast. Figure 9.9 shows the fuzzy sets describing the above. The rules base, with its 20 rules, is shown below. Figure 9.10 shows the result of the simulation of the rules base by fuzzy inference development environment (FIDE) software. Do you think any of the rules or fuzzy set limits should be modified?

```
If slope is Large-Positive and terrain is Very-Rough then speed is Very-Slow;
If slope is Large-Positive and terrain is Rough then speed is Slow;
If slope is Large-Positive and terrain is Moderate then speed is Medium;
If slope is Large-Positive and terrain is Smooth then speed is Medium;
If slope is Positive and terrain is Very-Rough then speed is Very-Slow;
If slope is Positive and terrain is Rough then speed is Slow;
If slope is Positive and terrain is Moderate then speed is Medium;
If slope is Positive and terrain is Smooth then speed is Fast;
If slope is Level and terrain is Very-Rough then speed is Slow;
If slope is Level and terrain is Rough then speed is Medium;
If slope is Level and terrain is Moderate then speed is Fast;
If slope is Level and terrain is Smooth then speed is very-Fast;
If slope is Negative and terrain is Very-Rough then speed is Very-Slow;
If slope is Negative and terrain is Rough then speed is Slow;
If slope is Negative and terrain is Moderate then speed is Medium;
If slope is Negative and terrain is Smooth then speed is Fast;
If slope is Large-Negative and terrain is Very-Rough then speed is Very-Slow;
If slope is Large-Negative and terrain is Rough then speed is Very-Slow;
If slope is Large-Negative and terrain is Moderate then speed is Slow;
If slope is Large-Negative and terrain is Smooth then speed is Medium;
```

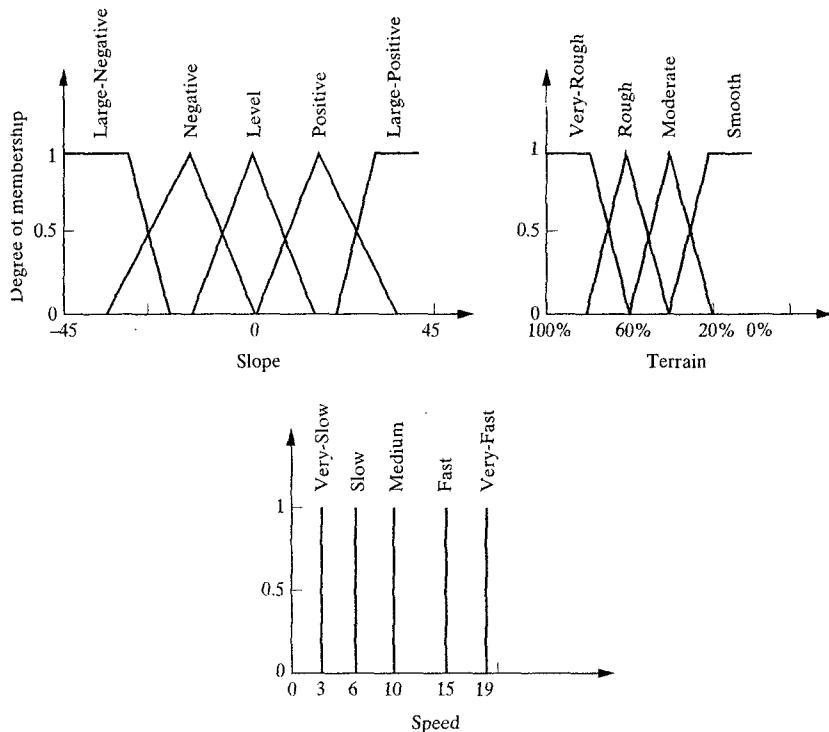


Figure 9.9 The fuzzy sets describing the inputs and the output of Example 9.2.

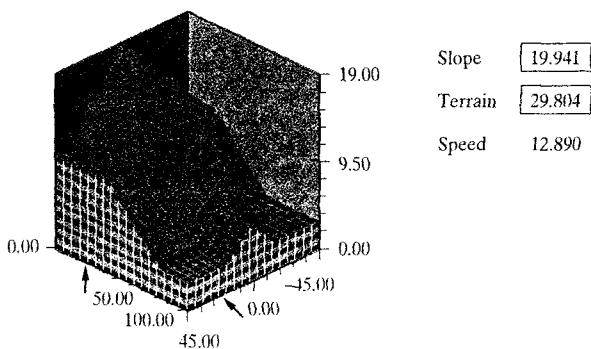


Figure 9.10 The result of the simulation of the system of Example 9.2 with FIDE software.

The surface shown in Figure 9.10 is the control surface. It means that for every possible value of the two inputs, there is a corresponding output based on the rules. For example, for the input values of Slope = 20 degrees and Terrain type = 30 (corresponding to half way between Smooth and Moderate) the output speed of the robot is about 13 miles per hour. As shown, the speed of the robot decreases as the slope increases up or down, as well as when the terrain becomes more and more rough.

Example 9.3

In a particular application, a robot is used to sort diamonds by weight and by color and then determine a price for the diamonds. Design a fuzzy logic system to control the process.

Solution Diamonds are classified by their weight, their color (indicated by letters, where A is extremely clear, while other letters indicates tints of yellow in the diamond), and by their purity (the size of inclusions). The clearer the diamond, the smaller the inclusions, and the larger the size, the more expensive the diamond will be per carat. In this example, we will assume that we are sorting the diamond by color and size (weight) only. We will 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 will also assume that using the techniques mentioned in Chapter 8, the vision system can recognize the diamond, its surface can be measured, and its weight can be estimated based on its size. We will further assume that the size of the diamonds will be in one of the sets of Small, Medium, Large, and Very-Large, as shown in Figure 9.11. The colors of the diamonds are divided into three color ranges of 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 \$100). The following shows the rule base, as well as the result of the simulation of this system by FIDE, shown in Figure 9.12.

```
If size is Small and color is D then price is Twenty;
If size is Medium and color is D then price is Thirty;
If size is Large and color is D then price is Forty;
If size is Very-Large and color is D then price is Fifty;
If size is small and color is H then price is Fifteen;
If size is Medium and color is H then price is Twenty;
If size is Large and color is H then price is Thirty;
If size is Very-Large and color is H then price is Forty;
If size is small and color is L then price is Ten;
If size is Medium and color is L then price is Fifteen;
If size is Large and color is L then price is Twenty;
If size is Very-Large and color is L then price is Thirty;
```

As can be seen, for every color and weight combination, there is a corresponding price. As an example, for a diamond of size 65 (relatively large), with a color indication of 56 (somewhere between D-H), the price indicated is 33.7 or \$3370 per carat. With this fuzzy logic system and only 12 rules, a vision system could estimate the corresponding price of diamonds automatically.

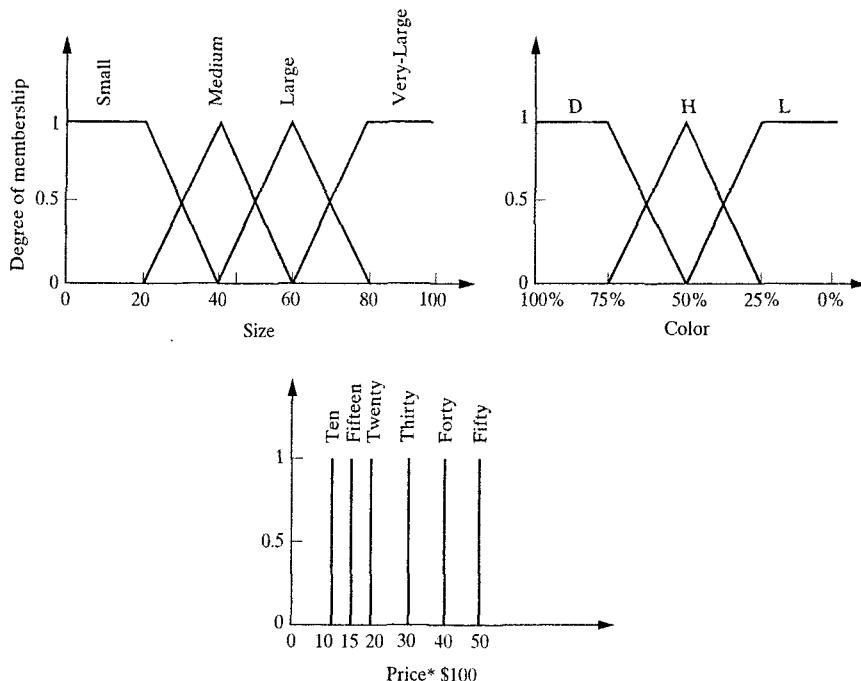


Figure 9.11 Fuzzy sets showing the input and the output of Example 9.3.

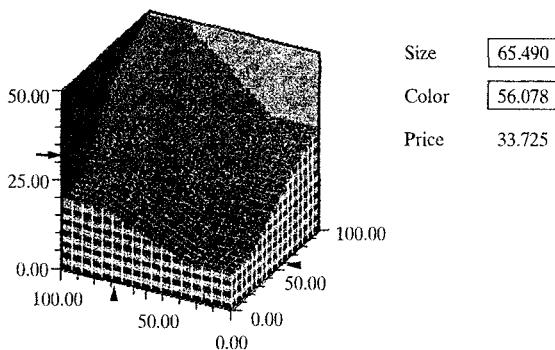


Figure 9.12 The result of the simulation of Example 9.3 with FLDE software.

9.10 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, for 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 it may be written for other purposes. For example, one may write a fuzzy control heuristics program such that the robot will follow a certain path based on fuzzy inputs. In addition, if you have access to a microprocessor, the developed programs can be downloaded to the microprocessor as part of the control program it runs. Please refer to the list of references for more information. Some of the books listed come with a fuzzy logic development software that you can use to simulate your designs.

9.11 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 nonexact concepts into everyday systems, including definitions (e.g., temperature and humidity), feelings (e.g., pain, hot, and cold), and adjectives (e.g., much and 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 control, to expert systems and vision systems. Fuzzy logic systems are generally simulated with a simulator program, and when it is verified that the system will behave 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. Fuzzy Inference Development Environment (FIDE) User's Manual, Aptronix Inc., San Jose, CA, 1992.
6. Fuzzy Design Generator (FUDGE), Motorola.
7. Fuzzy Knowledge Builder, McNeill, F. Martin, Ellen Thro.
8. 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.
9. 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

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 percent power to the actuators is the output.

2. Develop a fuzzy inference system for a washing machine. The inputs are how dirty the fabrics are and how much clothes are being washed, and the output is the wash time.
3. 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 or the time of cooking or both.
4. 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.
5. 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 colors) in a color image, and the output is the relationship of the combination to the colors of the rainbow.
6. 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 will be needed in our calculations are reviewed. You must already have an understanding of matrix algebra to understand the use of matrices. Thus, only a simple review of matrices is presented here.

Matrices 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 referred to as a square matrix.

Matrix Transpose The transpose of a matrix A_{ij}^T is another matrix A_{ji} , where elements of each row and column are replaced as follows:

$$A_{ij} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \text{ and } A_{ij}^T = A_{ji} = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{bmatrix}. \quad (0.1)$$

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:

$$C_{ij} = A_{ik} \times B_{kj} = \begin{bmatrix} d & e & f \\ g & h & l \end{bmatrix} \times \begin{bmatrix} p & s \\ q & t \\ r & w \end{bmatrix} = \begin{bmatrix} dp + eq + fr & ds + et + fw \\ gp + hq + lr & gs + ht + lw \end{bmatrix}. \quad (0.2)$$

As you see, the result of an $(m \times n)$ matrix multiplied by an $(n \times p)$ matrix is an $(m \times p)$ matrix, and thus the number of columns of the first matrix must be equal to the number of rows of the second matrix. Also, please 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 if 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 are not the same. 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:

$$A \times B \neq B \times A. \quad (0.3)$$

However,

$$A \times B \times C = (A \times B) \times C = A \times (B \times C), \quad (0.4)$$

$$(A + B)C = AC + BC \text{ and } C(A + B) = CA + CB. \quad (0.5)$$

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 unity, then the matrix is a unit matrix; premultiplying or postmultiplying any matrix by a unit matrix will result 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 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. Thus,

$$A_{ij} + B_{ij} = (A + B)_{ij}, \quad (0.6)$$

$$A + B + C = B + A + C = C + A + B \dots \quad (0.7)$$

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:

- Pick one row or column.
- 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 alternating plus and minus signs.

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. Then the determinant of the matrix will be

$$\det(A) = +a(ei - fh) - b(di - fg) + c(dh - eg).$$

Matrix Inversion This is an important operation in matrix representation of robots. We will be using matrix inversions for both inverse kinematics and for differential motions. In this section, two general-purpose inversion techniques for square matrices will be mentioned.

The inverse of a matrix is another matrix such that if the matrix is multiplied by the inverse, the result will be 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. Due to dimensional requirements, the left inverse and right inverse of a nonsquare matrix cannot be the same, even if they both exist. However, a square matrix will have the same left and right inverses, so 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- or post-multiplied by the square matrix, yielding a unit matrix.

Method 1 For square matrices with nonzero determinants only, the inverse of the matrix can be calculated by the following method:

- Calculate the determinant of the matrix.
- Transpose the matrix.
- Replace each element of the transposed matrix by its own minor (this is called an adjoint matrix).
- Divide the converted matrix (adjoint) by the determinant to get the inverse.

Thus,

$$A^{-1} = \frac{\text{adj}(A)}{\det(A)}. \quad (0.8)$$

The minor for each element A_{ij} of the matrix is the determinant of the matrix which 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 Equation (0.9). For example, we can write the following minors for the given matrix in Equation (0.9):

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, \quad \text{with} \quad \text{Sign} = \begin{bmatrix} + & - & + \\ - & + & - \\ + & - & + \end{bmatrix}, \quad (0.9)$$

where

$$a_{\text{minor}} = +(ei - fh),$$

$$b_{\text{minor}} = -(di - fg),$$

and

$$h_{\text{minor}} = -(af - cd), \dots$$

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 will calculate the determinant of the matrix:

$$\det(A) = 1(-1 + 8) - 0(0 - 20) + 1(0 - 5) = 7 - 5 = 2.$$

Next the transpose of the matrix will be formed:

$$A^T = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & -2 \\ 1 & 4 & -1 \end{bmatrix}.$$

Next, each element of the transposed matrix is replaced with its minor, yielding

$$A_{\text{minor}}^T = \begin{bmatrix} +(-1 + 8) & -(0 + 2) & +(0 - 1) \\ -(0 - 20) & +(-1 - 5) & -(4 - 0) \\ +(0 - 5) & -(-2 + 0) & +(1 - 0) \end{bmatrix} = \begin{bmatrix} 7 & -2 & -1 \\ 20 & -6 & -4 \\ -5 & 2 & 1 \end{bmatrix}.$$

The inverse can be found by dividing the matrix by the determinant as

$$A^{-1} = \begin{bmatrix} 3.5 & -1 & -0.5 \\ 10 & -3 & -2 \\ -2.5 & 1 & 0.5 \end{bmatrix}.$$

To ensure that the result is correct, you may multiply A by A^{-1} :

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 4 \\ 5 & -2 & -1 \end{bmatrix} \times \begin{bmatrix} 3.5 & -1 & -0.5 \\ 10 & -3 & -2 \\ -2.5 & 1 & 0.5 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Please verify that $A^{-1} \times A$ results in a unit matrix as well.

Method 2 In this method, we will assume an inverse matrix of the following form exists, such that when multiplied by the given matrix, a unit matrix results as in

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1i} \\ a_{21} & \cdot & & \\ \cdot & \cdot & & \\ a_{i1} & \cdot & & a_{ii} \end{bmatrix} \times \begin{bmatrix} x_{11} & x_{12} & \cdots & x_i \\ x_{21} & \cdot & & \\ \cdot & \cdot & & \\ x_{i1} & \cdot & & x_{ii} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (0.10)$$

where the x_{ii} matrix is the inverse of the A matrix that we are looking for. You should notice that since this represents a set of i equations and i unknowns, each one can be solved individually. If you multiply the first matrix A by the first column of the x matrix, the result will be the first column of the unit matrix. Then there will be a set of i equations that you will 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 preceding information, we will 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 notice, 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 trace A . Thus,

$$\text{trace } A = \sum_{j=1}^n a_{jj}.$$

Specifically, the trace for the product of a vector of n elements and its transpose is

$$\begin{aligned} \text{trace}[V \times V^T] &= \text{trace} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} \\ &= \text{trace} \begin{bmatrix} v_1^2 & v_1v_2 & \dots & v_1v_n \\ v_2v_1 & v_2^2 & & \\ \vdots & & & \\ v_nv_1 & & v_n^2 & \end{bmatrix} = \sum_{j=1}^n v_j^2. \end{aligned}$$

This will be used in the calculation of kinetic energy in Chapter 4.

Transpose of Products of Matrices The following is true:

$$\text{If } [B] \times [C] = [A], \text{ then } [C]^T \times [B]^T = [A]^T \quad (0.11)$$

For example, we can see that the following is true:

$$\begin{aligned} [a & b] \begin{bmatrix} c & d \\ e & f \end{bmatrix} &= [ac + be \quad ad + bf] \\ \begin{bmatrix} c & e \\ d & f \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} &= \begin{bmatrix} ac + be \\ ad + bf \end{bmatrix} = [ac + be \quad ad + bf]^T. \end{aligned}$$

A.2 CALCULATION OF AN ANGLE FROM ITS SINE, COSINE, OR TANGENT

There will be many situations in robotic calculations where the magnitude of the sine, cosine, or tangent of an angle is known, and we need to calculate the magnitude of the angle. Although this seems to be a trivial matter, in reality, it is very important, because there can be grave ambiguities in the answer that may yield an erroneous result, stopping a robot controller from functioning properly. This is true even with a calculator or a computer. To understand this, let's do a simple calculation.

Suppose that 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° angle. 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 thus the calculator always returns the smaller angle. The same is true for cosine and tangent of an angle; the cosine of the plus or minus of the same angle is the same, while the tangent of an angle is the same if 180° is added to it. This is simply demonstrated by the trigonometric relationships, as shown in Figure A.1.

To know the exact magnitude of an angle, it is necessary to determine in what quadrant the angle lies. This will enable 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.

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 sines and the cosine of 75° and 105° , we can easily calculate their correct values. The same principle is true for the tangent 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 sine and the cosine of the angle are found and used in a function, we can calculate the correct angle. FORTRAN computer language has a function called $\text{ATAN2}(\sin, \cos)$, 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 sign of the two, we will determine the quadrant and, thus, the correct value of the angle. This will be emphasized throughout this book whenever possible.

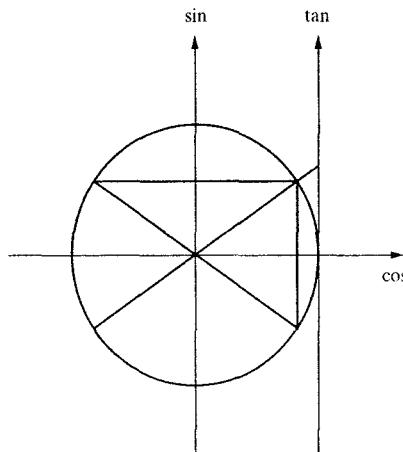


Figure A.1 Trigonometric functions.

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 = $\text{atan}(\alpha)$.
- If sin is positive and cos is negative, the angle is in quadrant 2, then angle = $180 - \text{atan}(\alpha)$.
- If sin is negative and cos is negative, the angle is in quadrant 3, then angle = $180 + \text{atan}(\alpha)$.
- If sin is negative and cos is positive, the angle is in quadrant 4, then angle = $-\text{atan}(\alpha)$.

The program should also check to see if either the sine or the cosine are 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.

PROBLEMS

1. Show that the determinant of a matrix can be calculated by picking any row or column.
2. Calculate the determinant of the following (4×4) matrix:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 3 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

3. Calculate the inverse of the following matrix using method 1:

$$B = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 3 \end{bmatrix}$$

4. Calculate the inverse of the following matrix using method 2:

$$C = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 3 & 1 & 0 \end{bmatrix}$$

Index

- A**
- Absolute encoders, 224
 - position, 225
 - Absolute encoder-strip, 227
 - Acceleration-deceleration regiment
 - trajectory planning, 152
 - Acceleration sensors, 230–231
 - Accuracy
 - sensor, 221
 - AC motors, 189
 - Actuating systems
 - characteristics, 174–177
 - comparison, 178
 - Actuators, 1–2, 7, 11, 15, 26, 55, 85–86, 116, 119, 132, 143–144, 151, 159, 170, 173–218, 231, 246, 328
 - exercises dealing with, 218
 - types, 173
 - ADC, 254
 - Adjoint, 50, 333–334
 - Adobe Photoshop, 306
 - Advantages, 5, 149, 215–216, 226, 246
 - AL, 17, 307
 - Aliasing
 - of sampled data, 264
 - Allegro Micro Systems, 205
 - AM120 Fanuc robot, 21
 - AML (A Manufacturing Language), 17, 19, 26
 - program example, 19–21

- Analog cameras, 250
- Analog-to-digital converter (ADC), 254
- Angular momentum, 128
- Anthropomorphic, 11–12, 54
- Arm, 24
- Articulated, 11–12, 16, 54, 59–60, 66–67
- Articulated coordinates, 59
- Articulated joints, 66
- Articulate robot
 - six degrees of freedom, 72
- Aspect ratio
 - of object, 290
- Assembly, 11–12, 20, 22, 196, 206, 216, 225, 305
 - operations, 22
- Assisting
 - disabled individuals, 23
- Association Francaise de Robotique (AFR), 3
- ATAN, ATAN2, 62, 337

B

- Back-emf, 190, 202
- Background pixels, 285
- Backlash, 175, 178
- Bifilar stepper motors, 200
- Binary code, 225–226
- Binary morphology operations, 284–287
- Biometal, 211
- Biometal-wire actuators, 212

- Bipolar drive circuits
 schematic drawing, 201
- Bipolar stepper motors, 200
- Bit, 208, 226, 262
- Brushless DC motors, 189
- Byte, 208
- C**
- C-programming, 196
- Calipers, 198
- Canstack motors, 193–197, 301–302
- Canstack rotor
 schematic, 194
- Canstack stator, 194
 windings and shells, 194
- Canstack stepper motor
 cross section, 196
- Capacitive proximity sensors, 237–238
- Capek, 4
- CAQ, 303
- Cartesian, 8, 11–12, 16, 29, 54–55, 58, 67, 86, 90, 149, 151, 153, 165, 171
 coordinator robot, 55
- Cartesian-space, 149
 descriptions
 vs. joint-space, 148–149
 trajectories, 165–170
- Cathode-ray tube (CRT), 251
- CCD. See Digital camera
- Center-of-gravity method, 318
- Centrifugal terms, 133
- Centripetal accelerations, 124
- Characteristics, 15–16, 22, 36, 153
- Charge coupled device (CCD). See also Digital camera
 VHS camera, 253
- Charge integrated device (CID) camera. See
 Digital camera
- Cincinnati Milacron, 178
- Classification, 2–3
- Close operation, 287
- Color images, 252
- Combined transformations
 representation, 43–46
- Compiler, 17–18
- Compiler based, 17–18
- Compliance, 178
 vs. stiffness, 174–175
- Components, 6–7, 10, 20, 22, 25–26, 32–34, 39, 57, 59, 62, 66, 89, 96, 106, 108, 118, 120, 131, 178, 180, 215, 271, 282
- Computed tomography (CT), 297–298
- Connectivity, 269–271
- CONSIGHT, 302
- Constant-area quantization (CAQ), 303
- Continuous trajectory recording, 170
- Continuous walk-through mode, 14–15
- Control device
 proportional feedback
 schematic, 183
- Controllers, 7–8, 13, 220
- Convolution mask, 259–262, 272–273
- Coordinates, 7–8, 11, 17, 29, 31–32, 40–47, 54–64, 67, 72, 89–90, 122, 149, 152–153, 168
- Coriolis accelerations, 124, 133, 134
- Cost
 sensor, 220
- Crisp rule, 314
- Crisp values
 vs. fuzzy values, 314
- CRT, 251
- CT, 297–298
- Cube in space
 represents, 35
- Current, 7, 19, 43, 46–48, 60–61, 64–65, 70, 85, 89, 95, 104–105, 110
- Cylindrical, 11–12, 16, 29, 54, 56–58, 63–64, 67, 86, 118, 189, 216
- coordinate system, 56–57
- rover schematic, 216
- D**
- DC motors, 188–189
 direction control, 210
- Defuzzification, 318–322
- Degeneracy, 82–83
- Degenerate position
 example of robot, 82
- Degrees of freedom (DOF), 2, 6, 8–11, 29, 36, 52–54, 72, 82, 86, 90, 95, 97, 111, 116, 121–122, 124, 144
- manipulator-type robots, 29–30
- multiple
 dynamic equations, 128–138
- one, 30, 121

- robot, 8–11
 - three, 85–86, 115–116, 168
 - two, 96, 122, 126, 136, 150–151, 167
 - joints-space, 150–151
 - Denavit Hartenberg representation, 29
 - forward kinematic equations of robots, 67–75
 - fundamental problem, 83–84
 - of general-purpose joint-link combination, 69
 - transformation matrices, 129
 - Depth analysis, 299
 - Depth measurement
 - vision systems, 298–301
 - Design project
 - rolling-cylinder robot rover, 215
 - three-degree-of-freedom robot, 85–86, 115–116, 143, 170–171, 245–246, 306, 327–328
 - Design project 1, 215
 - Detent torque, 202
 - Devol, George, 4
 - Dexterity, 6, 82–83
 - DFT, 297
 - D-H representation. See Denavit Hartenberg representation
 - Diagonal mask, 278–279
 - Dielectric constants, 238
 - Differential change
 - between frames, 104–106
 - interpretation, 104
 - Differential dithering, 185
 - Differential motions, 95–118
 - exercises dealing with, 117–118
 - of frame, 99–103
 - of robot and hand frame, 106–107
 - Differential operator
 - Jacobian, 110–111
 - Differential relationships, 95–97
 - Differential rotations, 100–101
 - about general axis K, 101–102
 - Differential transformations
 - of frame, 102–103
 - Differential translations, 100
 - Differential velocities, 95–118
 - exercises dealing with, 117–118
 - Digital camera, 250, 252
 - image acquisition, 253
 - Digital devices, 208
 - Digital images, 254
 - Digital-to-analog converter (DAC), 220
 - Digitized signal, 257
 - Dilation, 285, 288
 - Diode
 - stepper motors, 203
 - Direct drive, 173
 - electric motors, 189–190
 - Directional vector, 33
 - Disabled, 4, 23, 242
 - Disadvantages, 5, 149, 215–216, 226, 246
 - Discrete fourier transform (DFT), 297
 - Discrete signal, 257–258
 - Disk motor
 - schematic, 189
 - Dissipation
 - of heat, 187
 - DOF. See Degrees of freedom (DOF)
 - Dynamic analysis, 119–146
 - exercises dealing with, 144–146
 - Dynamic equations
 - for multiple-degree-of-freedom robots, 128–138
- E**
- Economy, 219, 248
 - Eddy current proximity sensors, 238
 - Edge detection, 275–279
 - Edges
 - of image, 257–258
 - Effective moments
 - of inertia, 127–128
 - Electric, 4, 7, 11, 173–176, 178–184, 186–191, 193, 195, 197, 199
 - actuator characteristics, 178
 - Electric motors, 186–207
 - microprocessor control, 207–210
 - Electromotive force, 190, 202
 - Encoders, 191, 219, 222–226, 230, 234, 236
 - incremental and absolute, 223–224
 - End effector, 6–7, 10–11, 14, 18, 20, 29, 48–49, 52, 70–71, 73, 82, 211–212, 232, 243, 324
 - frames, 49
 - Equations of motion, 122, 126, 136
 - Erosion, 285, 288
 - Euler angles, 64
 - Euler rotations
 - about current axes, 65

- F
Fanuc, 1, 3, 5, 7, 9, 15, 21, 23
Fanuc M-41iWW, 7
Fanuc robot
 P-15, 9
 P200, 21
Fast Fourier Transform (FFT). See **Fourier transform**
Features
 object recognition, 288–298
Feedback, 7–8, 15, 86, 219, 222, 245–247
FFT. See **Fourier transform**
Fifth-order polynomial trajectory planning, 157
Fill operation, 287
Filtering techniques, 272
Fixed-reference frame
 representation, 33–34
Fixed-sequence robot, 3
Flex sensors, 235
FM, 251
Force. 8, 67, 119–146, 149, 231–235, 238, 247
 exercises dealing with, 144–146
Force-acceleration relationship, 121
Force control
 of robots, 140
Force-sensing resistor (FSR), 231, 232
Force sensors, 231–232
Foreground pixels, 285
Foreshortening gradient, 301
Forward kinematic, 29, 53–54
 equations of robots
 Denavit Hartenberg representation, 67–76
Fourier series, 255
Fourier transform, 255–256, 258
Frame
 differential changes, 104–106
 differential motions, 99–103
 differential transformations, 102–103
Frame at the origin
 representation, 33–34
Frame in a fixed reference frame
 representation, 34–35
Frame in a frame
 representation, 34
Frame in space
 example, 35
Frequencies, 256
Frequency content
 of image, 257–258
 signal, 255–256
Frequency domain, 274
 plots, 255
 vs. spatial domain, 254
Frequency modulated (FM), 251
Frequency-related techniques, 272
Frequency response
 sensor, 221
Frequency spectrum, 254–258, 262, 265, 272, 297
FSR, 231, 232
Fuzzification, 315–316
Fuzzy control, 313
Fuzzy inference rule base, 316–317
Fuzzy logic control, 311–329
 description, 311–313
 exercises dealing with, 328–329
Fuzzy logic controller
 simulation, 322–323
Fuzzy logic in robotics
 applications, 323–327
Fuzzy logic system
 to sort diamonds, 326–327
Fuzzy sets
 degrees of membership and truth, 314
 temperature variable, 316
Fuzzy value, 314
- G
Gantry, 8, 11–12, 54–55, 183
Gantry Cartesian robot, 55
Gantry robot, 55
Gaussian averaging filters, 273
Gaussian elimination, 80
Gear reduction. See **reduction gears**
Gray code, 225–226
Gray intensity creation
 printed images, 250
Gray morphology operations, 288
Grayness values, 268
- H
Harmonic Drives, 213
Harmonic drive technologies, 214
Harmonics, 256. See also **Frequencies**
Hazardous, 5, 23, 26, 178
Hazardous environments, 23
H-bridge
 direction control, 210

- Heat dissipation path
of motors, 187
- Heuristics, 304–305
- Higher order trajectories, 161–165
- High-frequency signal, 257
- High-pass edge detector mask, 276
- High-pass filters, 276
- Histogram
of images, 267
- Histogram equalization
image, 268
- History, 4
- Holding torque, 201
- Homogeneous transformation matrices, 38
- Homogenous, 51
- Horizontal mask, 278–279
- Hough transform, 279–282
- Hybrid stepper motors, 198
operation, 199
- Hydraulic, 7–8, 11, 55, 173–176, 178–185, 217
actuators, 178–184
characteristics, 178
muscle type
schematic, 185
- Hydraulic system
components, 179–180
proportional feedback
block diagram, 183
schematic, 180
- Hysteresis, 187
- I**
- IBM 7565 linear hydraulic motor, 184
- IBM 7565 magneto reflective position sensor,
229
- IBM 7565 robot, 55, 185
- IBM's AML (A Manufacturing Language),
17, 26
program example, 19–20
- Image
acquisition, 250
defined, 249–250
histogram, 267
splitting, 282–283
- Image acquisition
digital camera, 253
- Image analysis, 288
- Image averaging, 273–274
- Image data collection model, 253
- Image data compression, 302–304
- Image processing
vs. image analysis, 248–249
techniques, 267
- Incremental encoders, 223–224
output signals, 225
wheel, 227
- Indexer, 192, 203, 207
- Inductive proximity sensors, 237
- Inertia
effective moments, 127–128
between motor and load, 176
- Infrared sensors, 233–234
- Injuries, 6
- Inspection, 21–22, 90, 117, 249, 305
- Integrator, 181
- Intelligent, 3, 5
- Intelligent robot, 3
- Interfacing
sensor, 220
- Interframe coding techniques, 303–304
- Interpretation
differential change, 104
- Interpreter, 17–18
- Interpreter based, 17–18
- Intraframe spatial domain techniques,
302–303
- Inverse Jacobian, 111–115
- Inverse kinematic equations, 58–59
to calculate joint velocities, 112
- Inverse kinematic programming of robots,
80–82
- Inverse kinematics, 29
- Inverse kinematic solution
for RPY, 62
- Inverse kinematic solution of robots, 76–80
- Inverse of a matrix
steps to calculate, 50
- Inverse of transformation matrices, 48–53
- J**
- Jacobian, 97–98
calculation of, 107–110
and differential operator, 110–111
of spherical-RPY robot, 140
- Joints, 67–70, 85
forces
to forces and moments, 139
offset, 70

- Joints (*continued*)
 reference frame, 13, 14
 types, 11
- Joint-space
 vs. Cartesian-space descriptions, 148–149
 description, 148
 trajectory planning, 153–165
 two-degree-of-freedom, 150–151
- Joint twist, 70
- Joint values
 joint-space trajectory planning, 153–155
- Joint variables, 80
- K**
- Kinematic equations
 forward and inverse
 for orientation, 59–60, 67
 for position, 54, 67
- Kinematics of robots
 forward and inverse, 53–67
- Kinetic energy, 128–132
- L**
- Lagrangian, 133
- Lagrangian mechanisms, 120–127
- Languages, 8, 16–19, 26, 306, 337
- Languages
 robot, 16–20
- Laplacian1, 276
 edge detectors, 277
- Laplacian2, 276
 edge detectors, 277
- Lapsed time, 239
- Laser-based range finders, 240–241
- Lead through mode, 13
- Left-right search technique
 edge detection, 278
- Light-based range finders, 240–241
- Light sensors, 233–234
- Linear
 joints, 11
- Linear hydraulic motor, 185
- Linear incremental encoder, 223
- Linearity
 sensor, 220–221
- Linear movements, 55
- Linear segments
 with parabolic blends, 157–160
- scheme, 158
 via points, 160–161
- Linear variable differential transformers
 (LVDT), 226–228
- Low-pass filter, 257–258
- LVDT, 226–228
- M**
- Machine, 4–5, 8, 17, 20, 87, 139, 220, 236, 242, 307–308, 312–314, 323, 328–329
- Machine loading, 20
- Magnetic proximity sensors, 236
- Magneto reflective displacement sensor, 229
- Magnetostrictive actuators, 210–211
- Mamdani's inference method, 318–319
 application, 322
- Manipulator, 1–2, 6–7, 9, 24, 29, 77, 132, 172
- Manipulator-type robots
 degrees of freedom (DOF), 29
- Manufacturing, 22
- Mapping
 vs. scene analysis, 298–299
- Mark II robot, 5
- Matrix representation, 31–38
- Maximum torque, 176–177
- Max operator, 288
- Median filters, 274–275
- Medical, 5, 23, 25, 242
- Medical applications, 23
- MEMS, 25, 27
- Microcomputer machine language level, 17
- Microcontroller
 schematic drawing, 204
- Micro-Electro-Mechanical-Systems (MEMS), 25, 27
- Microprocessor, 196, 203, 205–207, 209–210, 217, 220
 control
 electric motors, 207–210
 defined, 208
- Microstepping stepper motors, 202
- Microswitches, 233
- Min operator, 288
- Moment equations, 294
- Moment of inertia, 175–177, 292–295
 Effective, 127
- Moments, 290–297
 image
 calculation, 291

- Moments between coordinate frames, 141–142
Moments.mac program, 296–297
Moments.macro, 295
 Morphology operations, 284
 Motion control
 of mobile robot, 324
 Motion in three dimensions, 128–132
 Motorola, 205
 Multiple-degree-of-freedom robots
 dynamic equations, 128–138
 Multiplication, 40, 100–101, 105
 Muscle type
 hydraulic actuators
 schematic, 185
 MVS909, 306
- N**
 Neighborhood averaging, 272–273
 Newtonian mechanics, 120
 Noise
 of image, 257–258
 Noise reduction, 271–275
 Numerical control robot, 3
 Nutating gear train concept, 213
 Nyquist rate, 302
- O**
 Object identification
 basic features, 289–290
 Object in space
 representation, 35
 Object recognition
 features, 288–298
 Odetics, 24
 One-degree-of-freedom
 closed-loop four-bar mechanism, 30
 system, 121
 Open-loop mechanisms, 30
 Open-loop robots
 deflection in link, 30
 Open operation, 287
 Optical encoders, 224
 Optical proximity sensors, 236–237
 Optimas 6.2 vision software, 295
 Optimas 6.2 vision systems, 306
 Orbidrive, 212
 Output
 sensor, 220
- P**
 3P, 11–12
 Painting, 21
 Parabolic blends
 linear segments, 157–160
 via points, 160–161
 Path
 vs. trajectory, 147
 Payload, 15–16, 31, 178, 217, 220
 PCM, 302–303
 P200 Fanuc robot, 21
 Physical setup, 13
 Pick and place, 19–20, 22, 178
 operations, 20
 Piezoelectric, 231
 Pixels, 252, 263, 282
 of image, 260
 Pixels number, 268
 Planetary gear train
 schematic, 213
 Playback, 3–4, 14, 170
 Playback robot, 3
 Pneumatic, 7, 10–11
 actuator characteristics, 178
 devices, 184–186
 Point in space
 representation, 31
 Point-to-point, 3, 17, 170
 Point-to-point level, 17
 Polynomials
 3rd order, 154
 5th order, 157
 higher order, 161
 Position analysis, 29–94
 exercises dealing with, 88–94
 Position sensors, 222–229
 Position signal
 differentiation, 230
 Potential energy, 123, 132
 Potentiometers, 222–223
 Precision, 5, 15, 23, 178, 246
 Predictive coding, 303
 Pressure sensors, 231–232
 Prewitt edge detectors, 277
 Primitive motion level, 17–18
 Prismatic, 11, 13, 19, 49, 67–68, 74, 76, 82, 109,
 130, 139, 141
 joints, 11
 Processor, 8
 Programmable, 3, 5, 13

- Programming modes, 13–15
 Proximity sensors, 236
 Pulse code modulation (PCM), 302–303
 Pulse width modulation (PWM), 209–210
 PUMA, 5, 18, 92–93
 Pure rotation about an axis
 representation, 40–43
 Pure translation
 representation, 39–40
 Pusedorandom quantization dithering, 302
 PWM, 209–210
- Q**
 Quantization, 262–263
- R**
 3R, 11–12
 Rabota, 4
 Range detection, 299
 Range finders, 238–241
 Range measurement, 299
 triangulation, 239
 Range sensor, 221
 Raster scan depiction
 vidicon camera, 252
 RCC device, 242–245
 schematic, 245
 Reach, 14–16, 23, 82, 149, 155–156, 221
 Real time, 170, 304, 323
 image processing, 304
 Reconstruction of signals
 from sampled data, 266
 Rectangular, 5, 11–12, 38, 54–55
 Red, green, and blue (RGB), 252
 Reduction gears
 use of, 175–177
 Reference, 12, 114, 217, 231, 238, 294
 Reference frame, 9, 13–14, 19, 31–36, 38–49,
 54–61, 64, 67–68, 70, 73, 88–89, 95,
 99, 104, 110, 117, 132, 141–143, 146,
 165
 hand frame of robot, 54
 six-degree-of-freedom articulate robot, 72
 Region growing, 282–283
 Region splitting, 282
 Reliability
 sensor, 221
 Remote center compliance (RCC) device, 242–
 245
 Repeatability, 15–16, 221–222
 sensor, 221–222
 Resistor
 stepper motors, 203
 Resolution
 sensor, 220
 Resolver, 228
 schematic, 228
 Response time
 sensor, 221
 Revolute, 11, 13, 19, 49, 67–68, 72–74, 76, 81–
 82, 106–110, 112, 114, 118, 179
 joints, 11
 RGB colors, 252
 RIA, 3
 Rigid body
 representation, 35
 three-dimensional motion and plane motion,
 128
 Robert's edge detectors, 277
 Robot
 advantages and disadvantages, 5–6
 applications, 20–25
 characteristics, 15–16
 classification, 2–3
 components, 6–8
 coordinates, 11–12
 defined, 2
 degrees of freedom, 8–11
 joints, 11
 kinematics, 29–94
 exercises dealing with, 88–94
 languages, 16–20
 as mechanisms, 29–31
 reference frames, 12–13
 workspace, 16
 Robotics
 defined, 4
 history, 4–5
 Robotics Institute of America (RIA), 3
 Robot's equations of motion, 133–138
 Roll, pitch, yaw (RPY)
 angles, 60–64
 movements, 60
 rotations, 60–61
 Rolling-cylinder robot rover
 design project, 215

- R**
- Rotary
 - hydraulic actuator, 179
 - incremental encoder wheel, 223–224
 - joints, 11
 - potentiometers, 223
 - Rotation
 - of frame relative to reference frame, 43
 - matrices, 51
 - Rotational motion, 123
 - Rotor
 - of hybrid stepper motor, 198
 - inertia, 176–177
 - Rover, 6, 215–216, 246, 308
 - R2P, 11–12
 - 2RP, 11–12
 - RPY angles, 60–64
 - RPY movements, 60
 - RPY rotations
 - about current axes, 61
 - RPY sequence of rotation, 60
 - Rules
 - graphical representation, 317
 - Rules base
 - development of, 320
 - graphical representation, 321
- S**
- Sampling, 22, 226, 262–265, 302
 - Sampling theorem, 263–266
 - Scale factor, 32–51
 - Scaling gradient, 301
 - SCARA, 11–12, 67
 - Scene analysis
 - vs. mapping, 298–299
 - with shading and sizes, 300–301
 - Search technique
 - region growing, 283
 - Segmentation, 282
 - by region growing and region splitting, 282–284
 - Selective Compliance Assembly Robot Arm (SCARA), 11–12, 67
 - Sensitivity
 - sensor, 220
 - Sensors, 1, 6–8, 7, 26, 49, 140, 180–181, 191, 208, 215–216, 219–247
 - characteristics, 219–222
 - position, 222–229
 - Sequential robot movements
 - in path, 148
 - Servomotor, 190–191
 - actuator, 177
 - controller schematic, 191
 - Servo valves, 179–184
 - Shakey, 5
 - Shape memory type metals, 211
 - Sine functions
 - time and frequency domain, 256
 - Singularities, 149
 - Six-degree-of-freedom, 24
 - articulate robot, 72
 - reference frames, 114
 - Six-degree-of-freedom articulate robot
 - reference frame, 72
 - Size
 - sensor, 220
 - Skeletonization, 286–287
 - Skinlike tactile sensor, 235
 - Sliding
 - joints, 11
 - Sniff sensors, 241
 - Sobel
 - edge detectors, 277
 - Social issues, 25
 - Software, 8
 - Software mode, 15
 - Space, 2, 4, 8–9, 15–16, 24, 26–27, 31–32, 35–36, 38–39, 53–54, 87–89, 116, 120, 135, 144, 148–149, 171, 174, 187, 189, 220, 279, 302–303, 305
 - Spatial-domain
 - vs. frequency domain, 254
 - operations, 259–262
 - techniques, 272
 - Specialized lighting, 301–302
 - Speed reduction, 212–214
 - Speed-torque curve, 202
 - Spherical, 11–13, 16, 29, 54, 57–58, 61–62, 67, 75, 86, 89–90, 118, 130, 184
 - coordinates, 57–59
 - Spool valve
 - schematic, 180, 182
 - Stanford Arm, 75–76
 - parameters table, 84
 - reference frames, 83–84
 - schematic drawing, 75
 - Static force analysis of robots, 139–141

- S**
 Staubli, 1, 5, 20–22, 93
 Staubli robot loading and unloading components, 20
 Staubli robots, 21
 Stepper motor, 191–207
 control, 203
 full step and half-step sequence, 196
 lead configurations, 201
 operation principle, 192–193
 speed-torque characteristics, 200
 structure, 192
 translator schematic, 205
 Stepper translators, 205
 Stereo imaging, 299–300
Stiffness
 vs. compliance, 174–175
 Strain gauges, 231, 232
 Structured, 18–19
 programming level, 18
 Surveillance, 22–23, 306
- T**
 TAC, 241
 Tachometers, 230
 Tactile sensors, 234–236
 Task-oriented level, 18
 Teach mode, 13
 Template matching, 297
 Thickening operations, 284
 Third-order polynomial trajectory planning, 154–156
 Three-degree-of-freedom robot, 168
 design project, 85–86, 115–117, 143, 170–171, 245–246, 306, 327–328
 Three-dimensional images, 249
 Three-dimensional machines, 30
 Three-dimensional open loop
 chain mechanisms, 29
 Three-fingered end effector, 212
 Thresholding, 268–269, 282
 Time-domain plots, 255
 Time of flight, 239
 Time-of-travel, 229
 displacement sensor schematic, 229
 Timer circuit
 schematic, 207
 Time-to-amplitude converter (TAC), 241
 Time-to-target, 156
 Tool, 13–14, 19, 49, 139
- Tool reference frame, 13, 14
Torque
 to forces and moments, 139
 relationship
 between motor and load, 176
 sensors, 233
 Touch sensors, 234–236
Trace
 matrices, 131–132, 336
 Trajectory
 vs. path, 147
 planning, 147–172
 acceleration-deceleration regimen, 152
 basics, 150–153
 exercises dealing with, 172
 Transformation, 5, 38–40, 42–52, 55–56, 58, 68, 71, 73, 75–76, 86, 89–90, 102–103, 117, 129–131, 141, 166–167, 279
 changing the order, 46
 defined, 38
 of forces, 141–142
 matrix, 55
 relative to rotating frame, 46–47
 representation, 38–39
 Translators, 205
 schematic, 206
 Triangulation, 238
 Two-degree-of-freedom
 joint space, 150–151
 planar mechanism, 96
 planar robot, 167
 robot arm, 126, 136
 system, 122
 Two-dimensional images, 249
 Two-link mechanism, 124
 Type A, 3
 Type B, 3
 Type C, 3
 Type D, 3
- U**
 Ultrasonic proximity, 237
 Ultrasonic range finders, 239
 Underwater, 24
 Unimate, 5
 Unimation, 5, 17–18, 26, 92
 Unipolar drive circuits
 schematic drawing, 201
 Unitary matrix, 50

Universe reference frame, 49
Unrotate, 58
Uranus flyby, 303

V

VAL, 17–18
Validity, 15
VAL-II
 program example, 18–19
Value
 of spherical-RPY robot, 140
Variability, 15–16
Variable-sequence robot, 3
Vector in space
 representation, 32–33
Velocity diagram, 96
Velocity sensors, 229–230
Vertical mask, 278–279
Vidicon camera, 250–252
 raster scan depiction, 252
 schematic, 251
Virtual work, 139–141
Vision camera
 analog and digital, 250

Vision systems, 241
 applications, 305–306
 depth measurement, 298–301
 image processing and analysis, 248–310

Voice-recognition devices, 241–242

Voice synthesizers, 242

W

Weight
 sensor, 220
Welding, 20
Western Space and Marine, 24
Wheatstone bridge, 232
Workspace, 2, 8, 16, 28, 82–83, 116
World, 4, 7, 13–14, 219, 308
World reference frame, 13, 14

Z

Zener diode
 stepper motors, 203

