Contents

Preface

Scientists often have the feeling that, through their work, they are learning about some aspect of themselves. Physicists see this connection in their work; as do, for example, psychologists and chemists. In the study of robotics, the connection between the field of study and ourselves is unusually obvious. Unlike a science that seeks only to analyze, robotics as currently pursued takes the engineering bent toward synthesis. Perhaps it is for these reasons that the field fascinates so many of us.

The study of robotics concerns itself with the desire to synthesize some aspects of human function through the use of mechanisms, sensors, actuators, and computers. Obviously, this is a huge undertaking, which seems certain to require a multitude of ideas from various "classical" fields.

Currently, different aspects of robotics research are carried out by experts in various fields. It is usually not the case that any single individual has the entire area of robotics in his or her grasp. A partitioning of the field is natural to expect. At a relatively high level of abstraction, splitting robotics into four major areas seems reasonable: mechanical manipulation, locomotion, computer vision, and artificial intelligence.

This book introduces the science and engineering of mechanical manipulation. This subdiscipline of robotics has its foundations in several classical fields. The major relevant fields are mechanics, control theory, and computer science. In this book, Chapters 1 through 8 cover topics from mechanical engineering and mathematics, Chapters 9 through 11 cover control-theoretical material, and Chapters 12 and 13 might be classed as computer-science material. Additionally, the book emphasizes computational aspects of the problems throughout; for example, each chapter that is concerned predominantly with mechanics has a brief section devoted to computational considerations.

This book evolved from class notes used to teach "Introduction to Robotics" at Stanford University during the autumns of 1983 through 1985. The first three editions have been used from 1986 to 2016. The fourth edition has benefited from this use, and incorporates corrections and improvements due to feedback from many sources. Thanks to all those who sent corrections to the author.

This book is appropriate for a senior undergraduate- or first-year graduate-level course. It is helpful if the student has had one basic course in statics and dynamics, a course in linear algebra, and can program in a high-level language. Additionally, it is helpful, though not absolutely necessary, that the student have completed an introductory course in control theory. One aim of the book is to present material in a simple, intuitive way. Specifically, the audience need not be strictly mechanical engineers, though much of the material is taken from that field. At Stanford, many electrical engineers, computer scientists, and mathematicians found the book quite readable.

Directly, this book is of use to those engineers developing robotic systems, but the material should be viewed as important background material for anyone who will be involved with robotics. In much the same way that software developers have usually studied at least some hardware, people not directly involved with the mechanics and control of robots should have some such background as that offered by this text.

Like the third edition, the fourth edition is organized into 13 chapters. The material will fit comfortably into an academic semester; teaching the material within an academic quarter will probably require the instructor to choose a couple of chapters to omit. Even at that pace, all of the topics cannot be covered in great depth. In some ways, the book is organized with this in mind; for example, most chapters present only one approach to solving the problem at hand. One of the challenges of writing this book has been in trying to do justice to the topics covered within the time constraints of usual teaching situations. One method employed to this end was to consider only material that directly affects the study of mechanical manipulation.

At the end of each chapter is a set of exercises. Each exercise has been assigned a difficulty factor, indicated in square brackets following the exercise's number. Difficulties vary between [00] and [50], where [00] is trivial and [50] is an unsolved research problem.[1] Of course, what one person finds difficult, another might find easy, so some readers may find the factors misleading in some cases. Nevertheless, an effort has been made to appraise the difficulty of the exercises.

At the end of each chapter, there is a programming assignment in which the student applies the subject matter of the corresponding chapter to a simple three-jointed planar manipulator. This simple manipulator is complex enough to demonstrate nearly all the principles of general manipulators without bogging the student down in too much complexity. Each programming assignment builds upon the previous ones, until, at the end of the course, the student has an entire library of manipulator software.

There are a total of 12 MATLAB exercises associated with Chapters 1 through 9. These exercises were developed by Prof. Robert L. Williams II of Ohio University, and we are greatly indebted to him for this contribution. These exercises can be used with the MATLAB Robotics Toolbox[2] created by Peter Corke, Principal Research Scientist with CSIRO in Australia.

Chapter 1 is an introduction to the field of robotics. It introduces some background material, a few fundamental ideas, the adopted notation of the book, and it previews the material in the later chapters.

Chapter 2 covers the mathematics used to describe positions and orientations in 3-space. This is extremely important material: By definition, mechanical manipulation concerns itself with moving objects (parts, tools, the robot itself) around in space. We need ways to describe these actions in a way that is easily understood and is as intuitive as possible.

Chapters 3 and 4 deal with the geometry of mechanical manipulators. They introduce the branch of mechanical engineering known as kinematics, the study of

[1] I have adopted the same scale as in *The Art of Computer Programming* by D. Knuth (Addison-Wesley).

[2] For the MATLAB Robotics Toolbox, go to http://petercorke.com/Robotics_Toolbox.html

motion without regard to the forces that cause it. In these chapters, we deal with the kinematics of manipulators, but restrict ourselves to static positioning problems.

Chapter 5 expands our investigation of kinematics to velocities and static forces.

In Chapter 6, we deal for the first time with the forces and moments required to cause motion of a manipulator. This is the problem of manipulator dynamics.

Chapter 7 is concerned with describing motions of the manipulator in terms of trajectories through space.

Chapter 8 many topics related to the mechanical design of a manipulator. For example, how many joints are appropriate, of what type should they be, and how should they be arranged?

In Chapters 9 and 10, we study methods of controlling a manipulator (usually with a computer) so that it will faithfully track a desired position trajectory through space. Chapter 9 restricts attention to linear control methods; Chapter 10 extends these considerations to the nonlinear realm.

Chapter 11 covers the field of active force control with a manipulator. That is, we discuss how to control the application of forces by the manipulator. This mode of control is important when the manipulator comes into contact with the environment around it, such as during the washing of a window with a sponge.

Chapter 12 overviews methods of programming robots, specifically the elements needed in a robot programming system, and the particular problems associated with programming industrial robots.

Chapter 13 introduces off-line simulation and programming systems, which represent the latest extension to the man–robot interface.

New in the 4th Edition:

- Additional exercises at the end of each chapter
- New section 8.9 on optical encoders
- New section 10.9 on adaptive control
- Updated material and references for changing technology
- Several new or updated figures
- More than 100 minor typos and other errors corrected

I would like to thank the many people who have contributed their time to helping me with this book. First, my thanks to the students of Stanford's ME219 in the autumn of 1983 through 1985, who suffered through the first drafts, found many errors, and provided many suggestions. Professor Bernard Roth has contributed in many ways, both through constructive criticism of the manuscript and by providing me with an environment in which to complete the first edition. At SILMA Inc., I enjoyed a stimulating environment, plus resources that aided in completing the second edition. Dr. Jeff Kerr wrote the first draft of Chapter 8. Prof. Robert L. Williams II contributed the MATLAB exercises found at the end of each chapter, and Peter Corke expanded his Robotics Toolbox to support this book's style of the Denavit–Hartenberg notation. I owe a debt to my previous mentors in robotics: Marc Raibert, Carl Ruoff, Tom Binford, and Bernard Roth.

Many others around Stanford, SILMA, Adept, and elsewhere have helped in various ways—my thanks to John Mark Agosta, Mike Ali, Lynn Balling, Al Barr,

Stephen Boyd, Chuck Buckley, Joel Burdick, Jim Callan, Brian Carlisle, Monique Craig, Subas Desa, Tri Dai Do, Karl Garcia, Ashitava Ghosal, Chris Goad, Ron Goldman, Bill Hamilton, Steve Holland, Peter Jackson, Eric Jacobs, Johann Jäger, Paul James, Jeff Kerr, Oussama Khatib, Jim Kramer, Dave Lowe, Jim Maples, Dave Marimont, Dave Meer, Kent Ohlund, Madhusudan Raghavan, Richard Roy, Ken Salisbury, Bruce Shimano, Donalda Speight, Bob Tilove, Sandy Wells, and Dave Williams.

I wish to thank Tom Robbins at Pearson for his guidance with the first and second editions.

The students of Prof. Roth's Robotics Class of 2002 at Stanford used the second edition and forwarded many reminders of the mistakes that needed to get fixed for the fourth edition.

Finally I wish to thank those helping with the fourth edition: Matt Marshall who contributed some new end of chapter exercises as well as other helpful feedback; and Julie Bai and Michelle Bayman at Pearson.

JJC

CHAPTER 1

Introduction

1.1 BACKGROUND

The history of industrial automation is characterized by periods of rapid change in popular methods. Either as a cause or, perhaps, an effect, such periods of change in automation techniques seem closely tied to world economics. Use of the **industrial robot**, which became identifiable as a unique device in the 1960s [1], along with computer-aided design (CAD) systems and computer-aided manufacturing (CAM) systems, characterizes the latest trends in the automation of the manufacturing process. These technologies are leading industrial automation through another transition, the scope of which is still unknown [2].

In North America, there was much adoption of robotic equipment in the early 1980s, followed by a brief pull-back in the late 1980s. Since that time, the market has been growing (see Fig. 1.1), although it is subject to economic swings, as are all markets.

Figure 1.2 shows the number of robots being installed per year worldwide. A major reason for the growth in the use of industrial robots is their declining cost and increasing abilities. By 2025 it is estimated that the average manufacturing employer will save 16% on labor by replacing human workers with robots. In some countries, it is even more favorable to employ robots (see Fig. 1.3). As robots become more cost effective at their jobs, and as human labor continues to become more expensive, more and more industrial jobs become candidates for robotic automation. This is the single most important trend propelling growth of the industrial robot market. A secondary trend is that, economics aside, as robots become more capable, they become *able* to do more and more tasks that might be dangerous or impossible for human workers to perform.

This book focuses on the mechanics and control of the most important form of the industrial robot, the **mechanical manipulator**. Exactly what constitutes an industrial robot is sometimes debated. Devices such as that shown in Fig. 1.4 are always included, while numerically controlled (NC) milling machines usually are not. The distinction lies somewhere in the sophistication of the programmability of the device; if a mechanical device can be programmed to perform a wide variety of applications, it is probably an industrial robot. Machines which are for the most part limited to one class of task are considered **fixed automation**. For the purposes of this

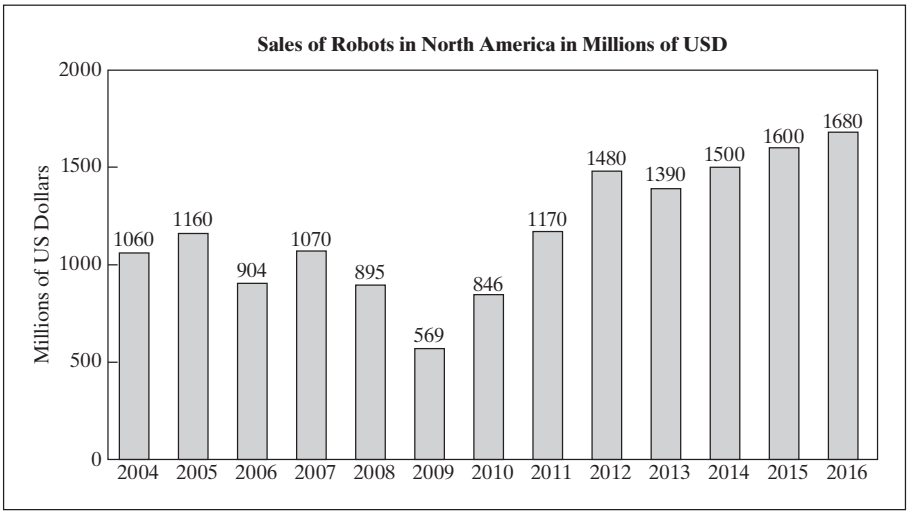

FIGURE 1.1: Sales of industrial robots in North America in millions of U.S. dollars. *Source:* Robotic Industries Association.

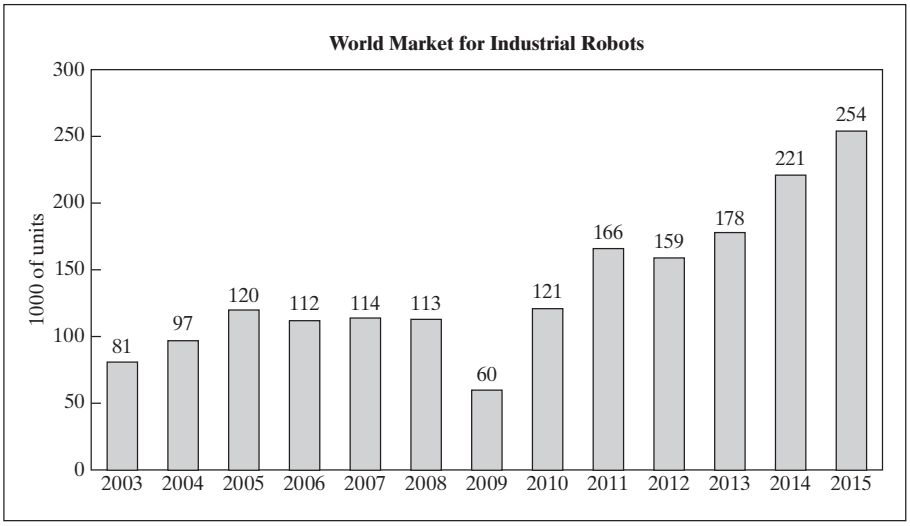

FIGURE 1.2: Yearly installations of multipurpose industrial robots. *Source:* World Robotics 2016.

text, the distinctions need not be debated; most material is of a basic nature that applies to a wide variety of programmable machines.

By and large, the study of the mechanics and control of manipulators is not a new science, but merely a collection of topics taken from "classical" fields. Mechanical engineering contributes methodologies for the study of machines in static and dynamic situations. Mathematics supplies tools for describing spatial motions and other attributes of manipulators. Control theory provides tools for designing and evaluating algorithms to realize desired motions or force applications.

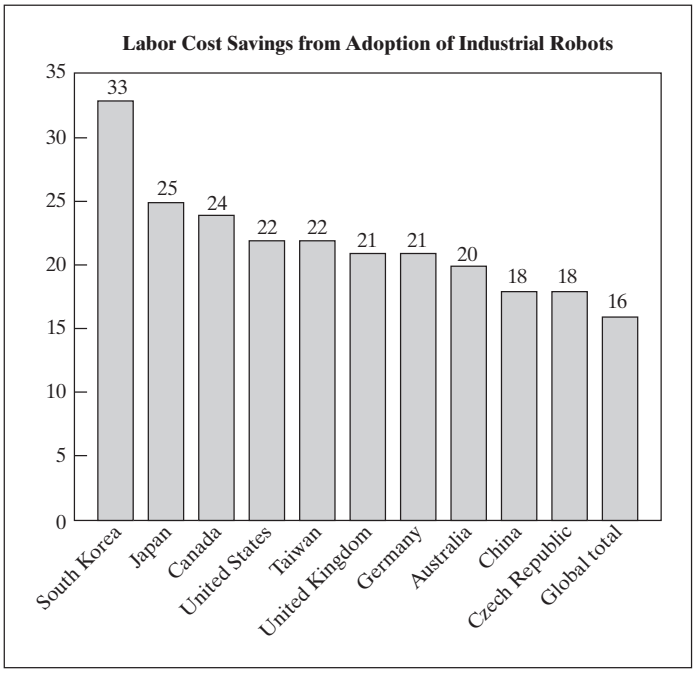

FIGURE 1.3: Labor cost savings from adoption of industrial robots. Estimated as a percentage in 2025. *Source:* The Boston Consulting Group.

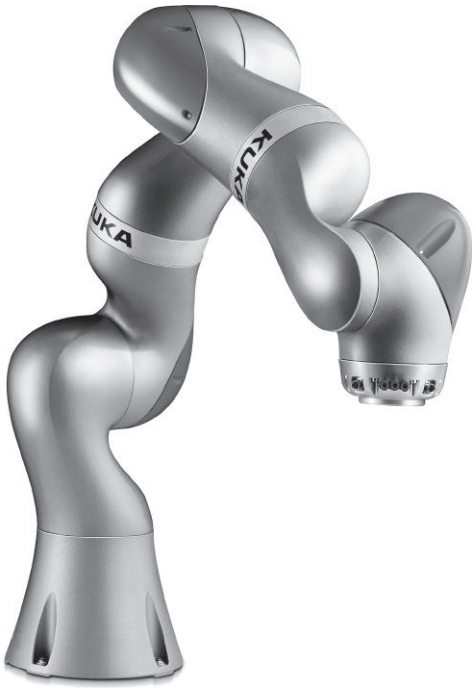

FIGURE 1.4: A modern 7 degree-of-freedom robot. Image courtesy KUKA Roboter GmbH.

Electrical-engineering techniques are brought to bear in the design of sensors and interfaces for industrial robots, and computer science contributes a basis for programming these devices to perform a desired task.

1.2 THE MECHANICS AND CONTROL OF MECHANICAL MANIPULATORS

The following sections will introduce some terminology, and briefly preview each of the topics that will be covered in the text.

Description of Position and Orientation

In the study of robotics, we are constantly concerned with the location of objects in three-dimensional space. These objects are the links of the manipulator, the parts and tools with which it deals, and other objects in the manipulator's environment. At a crude but important level, these objects are described by just two attributes: position and orientation. Naturally, one topic of immediate interest is the manner in which we represent these quantities and manipulate them mathematically.

In order to describe the position and orientation of a body in space, we will always attach a coordinate system, or **frame**, rigidly to the object. We will then proceed to describe the position and orientation of this frame with respect to some reference coordinate system (see Fig. 1.5).

Any frame can serve as a reference system within which to express the position and orientation of a body, so we often think of *transforming* or *changing the description of* these attributes of a body from one frame to another. Chapter 2 will discuss conventions and methodologies for dealing with the description of position and orientation, and the mathematics of manipulating these quantities with respect to various coordinate systems.

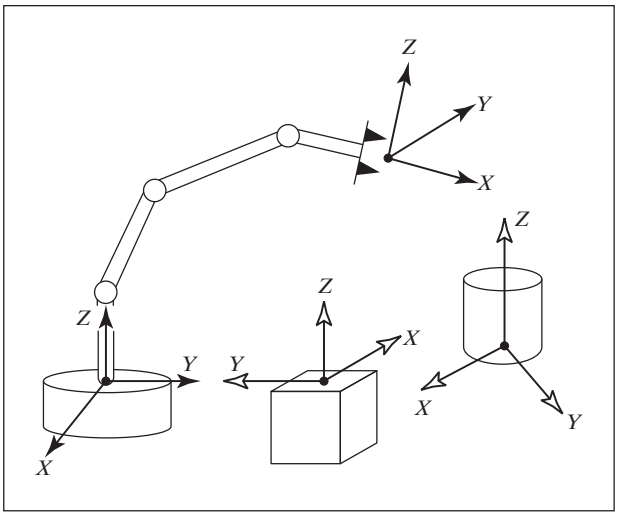

FIGURE 1.5: Coordinate systems or "frames" are attached to the manipulator and to objects in the environment.

Developing good skills concerning the description of position and rotation of rigid bodies is highly useful even in fields outside of robotics.

Forward Kinematics of Manipulators

Kinematics is the science of motion that treats motion without regard to the forces which cause it. Within the science of kinematics, one studies position, velocity, acceleration, and all higher order derivatives of the position variables (with respect to time or any other variable(s)). Hence, the study of the kinematics of manipulators refers to all the geometrical and time-based properties of the motion.

Manipulators consist of nearly rigid **links**, which are connected by **joints** that allow relative motion of neighboring links. These joints are usually instrumented with position sensors, which allow the relative position of neighboring links to be measured. In the case of rotary or **revolute** joints, these displacements are called **joint angles**. Some manipulators contain sliding (or **prismatic**) joints, in which the relative displacement between links is a translation, sometimes called the **joint offset**.

The number of **degrees of freedom** that a manipulator possesses is the number of independent position variables that would have to be specified in order to locate all parts of the mechanism. This is a general term used for any mechanism. For example, a four-bar linkage has only one degree of freedom (even though there are three moving members). In the case of typical industrial robots, because a manipulator is usually an open kinematic chain, and because each joint position is usually defined with a single variable, the number of joints equals the number of degrees of freedom.

At the free end of the chain of links that make up the manipulator is the **end-effector**. Depending on the intended application of the robot, the end-effector could be a gripper, a welding torch, an electromagnet, or another device. We generally describe the position of the manipulator by giving a description of the **tool frame**, which is attached to the end-effector, relative to the **base frame**, which is attached to the nonmoving base of the manipulator (see Fig. 1.6).

A very basic problem in the study of mechanical manipulation is called **forward kinematics**. This is the static geometrical problem of computing the position and orientation of the end-effector of the manipulator. Specifically, given a set of joint angles, the forward kinematic problem is to compute the position and orientation of the tool frame relative to the base frame. Sometimes, we think of this as changing the representation of manipulator position from a **joint space** description into a **Cartesian space** description.[1] This problem will be explored in Chapter 3.

Inverse Kinematics of Manipulators

In Chapter 4, we will consider the problem of **inverse kinematics.** This problem is posed as follows: Given the position and orientation of the end-effector of the manipulator, calculate all possible sets of joint angles that could be used to attain this given position and orientation (see Fig. 1.7). This is a fundamental problem in the practical use of manipulators.

[1] By *Cartesian space*, we mean the space in which the position of a point is given with three numbers, and in which the orientation of a body is given with three numbers. It is sometimes called *task space* or *operational space*.

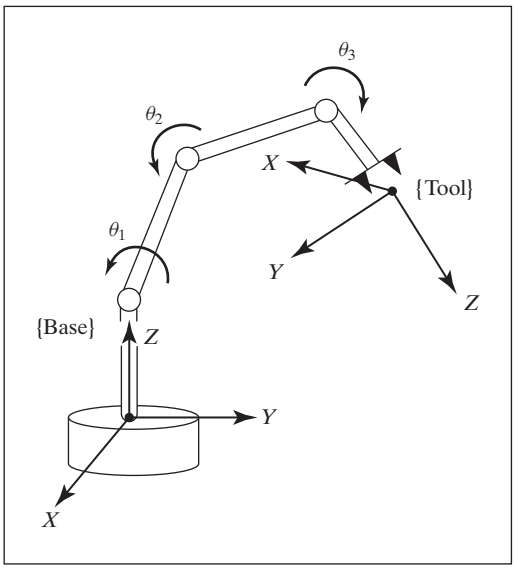

FIGURE 1.6: Kinematic equations describe the tool frame relative to the base frame as a function of the joint variables.

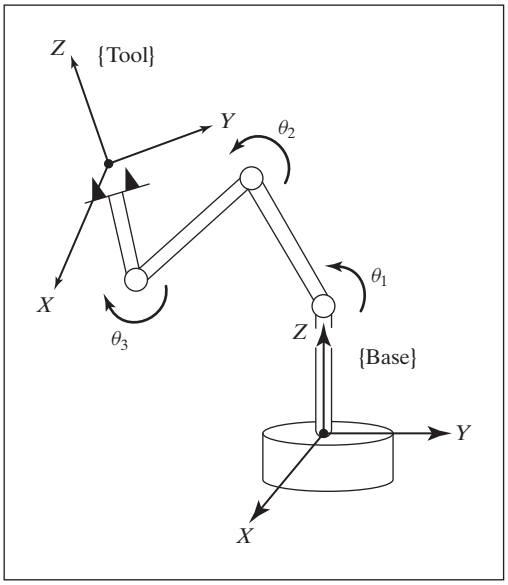

FIGURE 1.7: For a given position and orientation of the tool frame, values for the joint variables can be calculated via the inverse kinematics.

This is a rather complicated geometrical problem that is routinely solved thousands of times daily in human and other biological systems. In the case of an artificial system like a robot, we will need to create an algorithm in the control computer that can make this calculation. In some ways, solution of this problem is the most important element in a manipulator system.

We can think of this problem as a *mapping* of "locations" in 3-D Cartesian space to "locations" in the robot's internal joint space. This need naturally arises anytime a goal is specified in external 3-D space coordinates. Some early robots lacked this algorithm—they were simply moved (sometimes by hand) to desired locations, which were then recorded as a set of joint values (i.e., as a location in joint space) for later playback. Obviously, if the robot is used purely in the mode of recording and playback of joint locations and motions, no algorithm relating joint space to Cartesian space is needed. These days, however, it is rare to find an industrial robot that lacks this basic inverse kinematic algorithm.

The inverse kinematics problem is not as simple as the forward kinematics one. Because the kinematic equations are nonlinear, their solution is not always easy (or even possible) in a closed form. Also, questions about the existence of a solution and about multiple solutions arise.

Study of these issues gives one an appreciation for what the human mind and nervous system are accomplishing when we, seemingly without conscious thought, move and manipulate objects with our arms and hands.

The existence or nonexistence of a kinematic solution defines the **workspace** of a given manipulator. The lack of a solution means that the manipulator cannot attain the desired position and orientation, because it lies outside of the manipulator's workspace.

Velocities, Static Forces, Singularities

In addition to dealing with static positioning problems, we may wish to analyze manipulators in motion. Often, in performing velocity analysis of a mechanism, it is convenient to define a matrix quantity called the **Jacobian** of the manipulator. The Jacobian specifies a **mapping** from velocities in joint space to velocities in Cartesian space (see Fig. 1.8). The nature of this mapping changes as the configuration of the manipulator varies. At certain points, called **singularities**, this mapping is not invertible. An understanding of the phenomenon is important to designers and users of manipulators.

Consider the rear gunner in a World War I–vintage biplane fighter plane (illustrated in Fig. 1.9). While the pilot flies the plane from the front cockpit, the rear gunner's job is to shoot at enemy aircraft. To perform this task, his gun is mounted in a mechanism that rotates about two axes, the motions being called azimuth and elevation. Using these two motions (two degrees of freedom), the gunner can direct his stream of bullets in any direction he desires in the upper hemisphere.

An enemy plane is spotted at azimuth one o'clock and elevation 25 degrees! The gunner trains his stream of bullets on the enemy plane and tracks its motion so as to hit it with a continuous stream of bullets for as long as possible. He succeeds and thereby downs the enemy aircraft.

A second enemy plane is seen at azimuth one o'clock and elevation 70 degrees! The gunner orients his gun and begins firing. The enemy plane is moving so as to

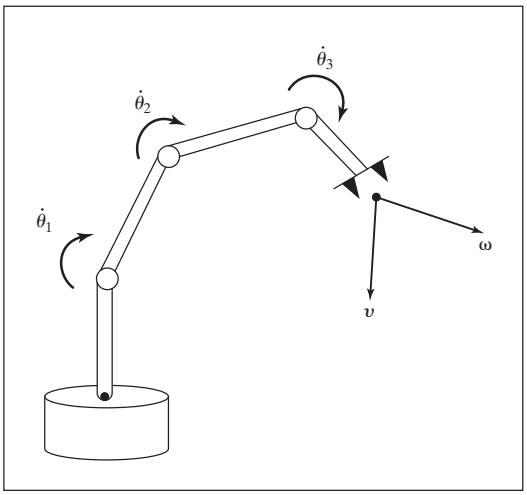

FIGURE 1.8: The geometrical relationship between joint rates and velocity of the end-effector can be described in a matrix called the Jacobian.

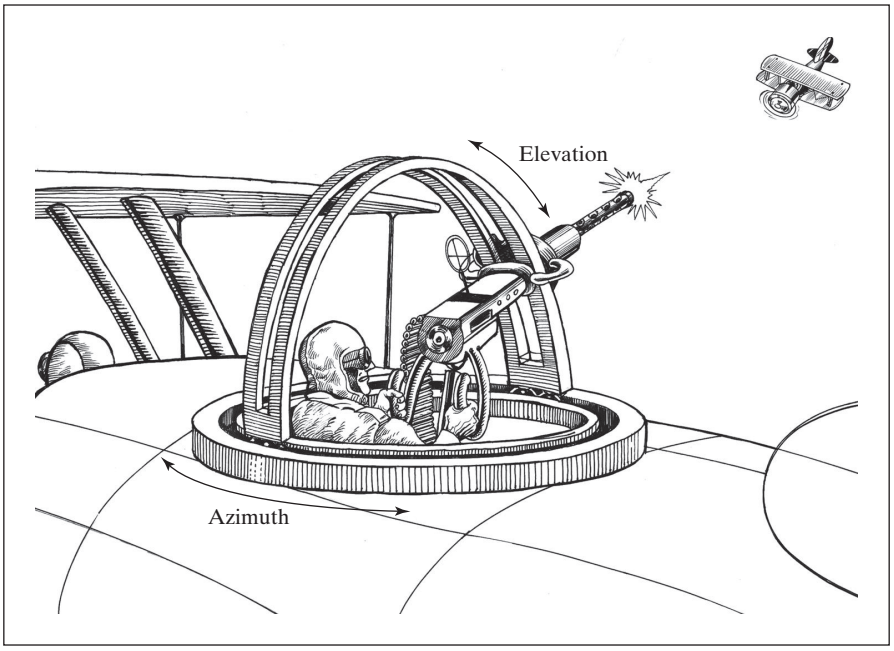

FIGURE 1.9: A World War I biplane with a pilot and a rear gunner. The rear-gunner mechanism is subject to the problem of singular positions.

obtain a higher and higher elevation relative to the gunner's plane. Soon the enemy plane is passing nearly overhead. What's this? The gunner is no longer able to keep his stream of bullets trained on the enemy plane! He found that, as the enemy plane flew overhead, he was required to change his azimuth at a very high rate. He was not able to swing his gun in azimuth quickly enough, and the enemy plane escaped!

In the latter scenario, the lucky enemy pilot was saved by a *singularity*! The gun's orienting mechanism, while working well over most of its operating range, becomes less than ideal when the gun is directed straight upwards or nearly so. To track targets that pass through the position directly overhead, a very fast motion around the azimuth axis is required. The closer the target passes to the point directly overhead, the faster the gunner must turn the azimuth axis to track the target. If the target flies directly over the gunner's head, he would have to spin the gun on its azimuth axis at infinite speed!

Should the gunner complain to the mechanism designer about this problem? Could a better mechanism be designed to avoid this problem? It turns out that you really can't avoid the problem very easily. In fact, any two-degree-of-freedom orienting mechanism that has exactly two rotational joints cannot avoid having this problem. In the case of this mechanism, with the stream of bullets directed straight up, their direction aligns with the axis of rotation of the azimuth rotation. This means that, at exactly this point, the azimuth rotation does not cause a change in the direction of the stream of bullets. We know we need two degrees of freedom to orient the stream of bullets, but, at this point, we have lost the effective use of one of the joints. Our mechanism has become **locally degenerate** at this location, and behaves as if it only has one degree of freedom (the elevation direction).

This kind of phenomenon is caused by what is called a **singularity of the mechanism**. All mechanisms are prone to these difficulties, including robots. Just as with the rear gunner's mechanism, these singularity conditions do not prevent a robot arm from positioning anywhere within its workspace. However, they can cause problems with *motions* of the arm in their neighborhood.

Manipulators do not always move through space; sometimes they are also required to touch a workpiece or work surface and apply a static force. In this case, the problem arises: Given a desired contact force and moment, what set of **joint torques** is required to generate them? Once again, the Jacobian matrix of the manipulator arises quite naturally in the solution of this problem.

Dynamics

Dynamics is a huge field of study devoted to studying the forces required to cause motion. In order to accelerate a manipulator from rest, glide at a constant end-effector velocity, and finally decelerate to a stop, a complex set of torque functions must be applied by the joint actuators.[2] The exact form of the required functions of actuator torque depend on the spatial and temporal attributes of the path taken by the end-effector and on the mass properties of the links and payload, friction in the joints, and so on. One method of controlling a manipulator to follow a desired path involves calculating these actuator torque functions by using the dynamic equations of motion of the manipulator.

[2] We use *joint actuators* as the generic term for devices that power a manipulator—for example, electric motors, hydraulic and pneumatic actuators, and muscles.

Many of us have experienced lifting an object that is actually much lighter than we expected (e.g., getting a container of milk from the refrigerator which we thought was full, but was nearly empty). Such a misjudgment of payload can cause an unusual lifting motion. This kind of observation indicates that the human control system is more sophisticated than a purely kinematic scheme. Rather, our manipulation control system makes use of knowledge of mass and other dynamic effects. Likewise, algorithms that we construct to control the motions of a robot manipulator should take dynamics into account.

A second use of the dynamic equations of motion is in **simulation**. By reformulating the dynamic equations so that acceleration is computed as a function of actuator torque, it is possible to simulate how a manipulator would move under application of a set of actuator torques (see Fig. 1.10). As computing power becomes more and more cost effective, the use of simulations is growing in use and importance in many fields.

In Chapter 6, we will develop dynamic equations of motion, which may be used to control or simulate the motion of manipulators.

Trajectory Generation

A common way of causing a manipulator to move from here to there in a smooth, controlled fashion is to cause each joint to move as specified by a smooth function of time. Commonly, each joint starts and ends its motion at the same time, so that the manipulator motion appears coordinated. Exactly how to compute these motion functions is the problem of **trajectory generation** (see Fig. 1.11).

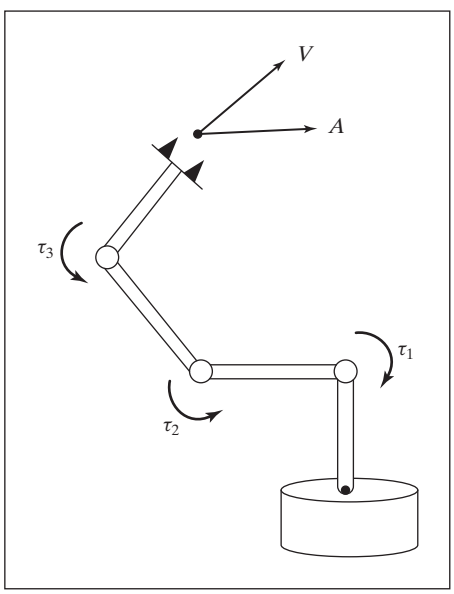

FIGURE 1.10: The relationship between the torques applied by the actuators and the resulting motion of the manipulator is embodied in the dynamic equations of motion.

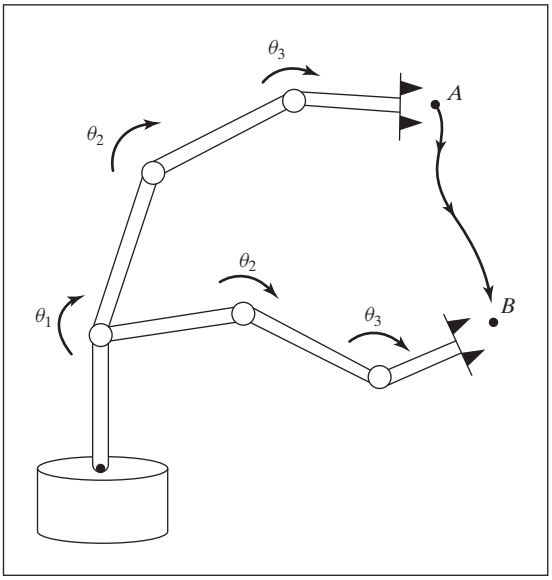

FIGURE 1.11: In order to move the end-effector through space from point A to point B, we must compute a trajectory for each joint to follow.

Often, a path is described not only by a desired destination but also by some intermediate locations, or **via points**, through which the manipulator must pass en route to the destination. In such instances, the term **spline** is sometimes used to refer to a smooth function that passes through a set of via points.

In order to force the end-effector to follow a straight line (or other geometric shape) through space, the desired motion must be converted to an equivalent set of joint motions. This **Cartesian trajectory generation** will also be considered in Chapter 7.

Manipulator Design and Sensors

Although manipulators are, in theory, universal devices applicable to many situations, economics generally dictate that the intended task domain influence the mechanical design of the manipulator. Along with issues such as size, speed, and load capability, the designer must also consider the number of joints and their geometric arrangement. These considerations affect the manipulator's workspace size and quality, the stiffness of the manipulator structure, and other attributes.

The more joints a robot arm contains, the more dextrous and capable it will be. Of course, it will also be harder and more expensive to build. In order to build a useful robot, that can take two approaches: build a **specialized robot** for a specific task, or build a **universal robot** that would able to perform a wide variety of tasks. In the case of a specialized robot, some careful thinking will yield a solution for how many joints are needed. For example, a specialized robot designed solely to place electronic components on a flat circuit board does not need to have more than four joints. Three joints allow the position of the hand to attain any position in

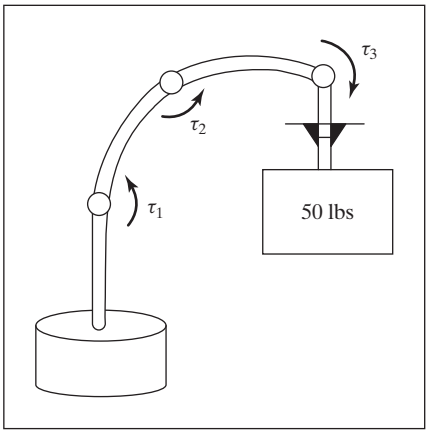

FIGURE 1.12: The design of a mechanical manipulator must address issues of actuator choice, location, transmission system, structural stiffness, sensor location, and more.

three-dimensional space, with a fourth joint added to allow the hand to rotate the grasped component about a vertical axis. In the case of a universal robot, it is interesting that fundamental properties of the physical world we live in dictate the "correct" minimum number of joints—that minimum number is six.

Integral to the design of the manipulator are issues involving the choice and location of actuators, transmission systems, and internal-position (and sometimes force) sensors (see Fig. 1.12). These and other design issues will be discussed in Chapter 8.

Linear Position Control

Some manipulators are equipped with stepper motors or other actuators that can directly execute a desired trajectory. However, the vast majority of manipulators are driven by actuators that supply a force or a torque to cause motion of the links. In this case, an algorithm is needed to compute torques that will cause the desired motion. The problem of dynamics is central to the design of such algorithms, but does not in itself constitute a solution. A primary concern of a **position control system** is to automatically compensate for errors in knowledge of the parameters of a system, and to suppress disturbances that tend to perturb the system from the desired trajectory. To accomplish this, position and velocity **sensors** are monitored by the **control algorithm**, which computes torque commands for the actuators (see Fig. 1.13). In Chapter 9, we will consider control algorithms whose synthesis is based on linear approximations to the dynamics of a manipulator. These linear methods are prevalent in current industrial practice.

Nonlinear Position Control

Although control systems based on approximate linear models are popular in current industrial robots, it is important to consider the complete nonlinear dynamics of the manipulator when synthesizing control algorithms. Some industrial robots

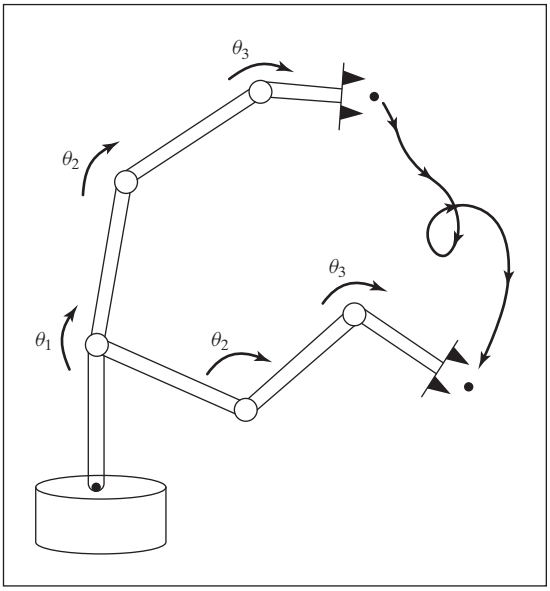

FIGURE 1.13: In order to cause the manipulator to follow the desired trajectory, a position-control system must be implemented. Such a system uses feedback from joint sensors to keep the manipulator on course.

are now being introduced which make use of **nonlinear control** algorithms in their controllers. These nonlinear techniques of controlling a manipulator promise better performance than do simpler linear schemes. Chapter 10 will introduce nonlinear control systems for mechanical manipulators.

Force Control

The ability of a manipulator to control forces of contact when it touches parts, tools, or work surfaces seems to be of great importance in applying manipulators to many real-world tasks. **Force control** is complementary to position control, in that we usually think of only one or the other as applicable in a certain situation. When a manipulator is moving in free space, only position control makes sense, because there is no surface to react against. When a manipulator is touching a rigid surface, however, position-control schemes can cause excessive forces to build up at the contact, or cause contact to be lost with the surface when it was desired for some application. Manipulators are rarely constrained by reaction surfaces in all directions simultaneously, so a mixed or **hybrid** control is required, with some directions controlled by a **position-control law** and remaining directions controlled by a **force-control law** (see Fig. 1.14). Chapter 11 introduces a methodology for implementing such a force-control scheme.

A robot should be instructed to wash a window by maintaining a certain force in the direction perpendicular to the plane of the glass, while following a motion trajectory in directions tangent to the plane. Such split or **hybrid** control specifications are natural for such tasks.

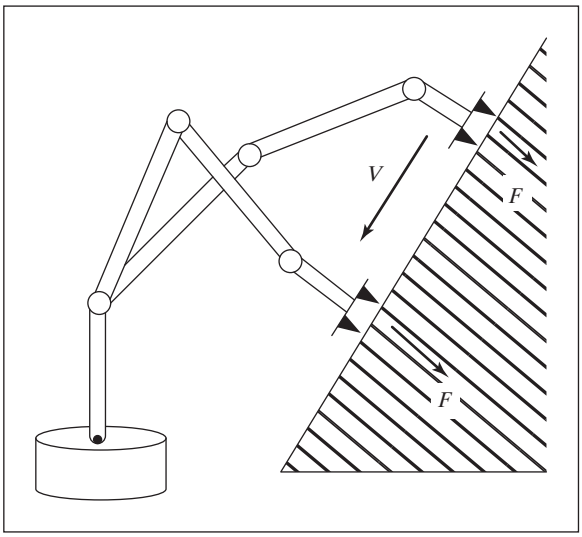

FIGURE 1.14: In order for a manipulator to slide across a surface while applying a constant force, a hybrid position–force control system must be used.

Programming Robots

A robot programming language serves as the interface between the human user and the industrial robot. Central questions arise: How are motions through space described easily by the programmer? How are multiple manipulators programmed so that they can work in parallel? How are sensor-based actions described in a language?

Robot manipulators differentiate themselves from **fixed automation** by being "flexible," which means programmable. Not only are the movements of manipulators programmable, but, through the use of sensors and communications with other factory automation, manipulators can *adapt* to variations as the task proceeds (see Fig. 1.15).

In typical robot systems, there is a shorthand way for a human user to instruct the robot which path it is to follow. First of all, a special point on the hand (or perhaps on a grasped tool) is specified by the user as the **operational point**, sometimes also called the **TCP** (for **Tool Center Point**). Motions of the robot will be described by the user in terms of desired locations of the operational point relative to a user-specified coordinate system. Generally, the user will define this reference coordinate system relative to the robot's base coordinate system in some task-relevant location.

Most often, paths are constructed by specifying a sequence of **via points**. Via points are specified relative to the reference coordinate system and denote locations along the path through which the TCP should pass. Along with specifying the via points, the user may also indicate that certain speeds of the TCP be used over various portions of the path. Sometimes, other modifiers can also be specified to affect the motion of the robot (e.g., different smoothness criteria, etc.). From these inputs, the trajectory-generation algorithm must plan all the details of the motion: velocity profiles for the joints, time duration of the move, and so on. Hence, input to the

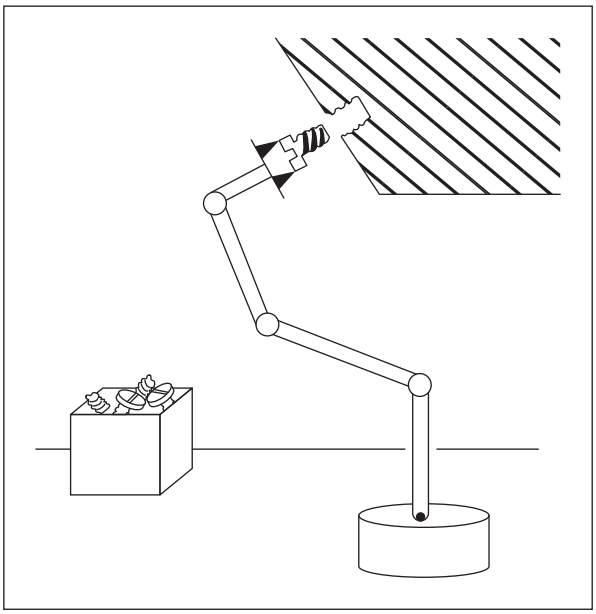

FIGURE 1.15: Desired motions of the manipulator and end-effector, desired contact forces, and complex manipulation strategies can be described in a *robot programming language*.

trajectory-generation problem is generally given by constructs in the robot programming language.

The sophistication of the user interface is becoming extremely important as manipulators and other programmable automation are applied to more and more demanding industrial applications. The problem of programming manipulators encompasses all the issues of "traditional" computer programming, and so is an extensive subject in itself. Additionally, some particular attributes of the manipulator-programming problem cause additional issues to arise. Some of these topics will be discussed in Chapter 12.

Off-Line Programming and Simulation

An **off-line programming system** is a robot programming environment that has been sufficiently extended, generally by means of computer graphics, that the development of robot programs can take place without access to the robot itself. A common argument raised in their favor is that an off-line programming system will not cause production equipment (i.e., the robot) to be tied up when it needs to be reprogrammed; hence, automated factories can stay in production mode a greater percentage of the time (see Fig. 1.16).

They also serve as a natural vehicle to tie computer-aided design (CAD) databases used in the design phase of a product to the actual manufacturing of the product. In some cases, this direct use of CAD data can dramatically reduce the

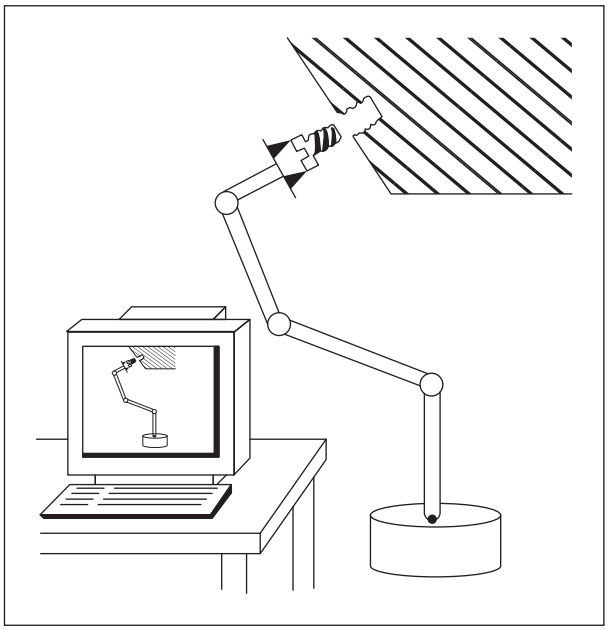

FIGURE 1.16: Off-line programming systems, generally providing a computer graphics interface, allow robots to be programmed without access to the robot itself during programming.

programming time required for the manufacturing process. Chapter 13 discusses the elements of industrial robot off-line programming systems.

1.3 NOTATION

Notation is always an issue in science and engineering. In this book, we use the following conventions:

1. Usually, variables written in uppercase represent vectors or matrices. Lowercase variables are scalars.
2. Leading subscripts and superscripts identify which coordinate system a quantity is written in. For example, ${}^{A}P$ represents a position vector written in coordinate system $\{A\}$, and ${}^{A}_{B}R$ is a rotation matrix[3] that specifies the relationship between coordinate systems $\{A\}$ and $\{B\}$.
3. Trailing superscripts are used (as widely accepted) for indicating the inverse or transpose of a matrix (e.g., R^{-1}, R^{T}).
4. Trailing subscripts are not subject to any strict convention, but may indicate a vector component (e.g., x, y, or z) or may be used as a description, as in P_{bolt}, the position of a bolt.
5. We will use many trigonometric functions. Our notation for the cosine of an angle θ_1 may take any of the following forms: $\cos\theta_1 = c\theta_1 = c_1$.

[3]This term will be introduced in Chapter 2.

Vectors are taken to be column vectors; hence, row vectors will have the transpose indicated explicitly.

A note on vector notation in general: Many mechanics texts treat vector quantities at a very abstract level and routinely use vectors defined relative to different coordinate systems in expressions. The clearest example is that of addition of vectors which are given or known relative to differing reference systems. This is often very convenient and leads to compact and somewhat elegant formulas. For example, consider the angular velocity, ${}^0\omega_4$, of the last body in a series connection of four rigid bodies (as in the links of a manipulator) relative to the fixed base of the chain. Because angular velocities sum vectorially, we may write a very simple vector equation for the angular velocity of the final link:

$$ {}^0\omega_4 = {}^0\omega_1 + {}^1\omega_2 + {}^2\omega_3 + {}^3\omega_4. \tag{1.1} $$

However, unless these quantities are expressed with respect to a common coordinate system, they cannot be summed, and so, though elegant, equation (1.1) has hidden much of the "work" of the computation. For the particular case of the study of mechanical manipulators, statements like that of (1.1) hide the chore of bookkeeping of coordinate systems, which is often the very idea that we need to deal with in practice.

Therefore, in this book, we carry frame-of-reference information in the notation for vectors, and we do not sum vectors unless they are in the same coordinate system. In this way, we derive expressions that solve the "bookkeeping" problem and can be applied directly to actual numerical computation.

BIBLIOGRAPHY

[1] B. Roth, "Principles of Automation," Future Directions in Manufacturing Technology, Based on the Unilever Research and Engineering Division Symposium held at Port Sunlight, April 1983, Published by Unilever Research, UK.

[2] R. Brooks, *Flesh and Machines*, Pantheon Books, New York, 2002.

[3] International Federation of Robotics, "Executive Summary World Robotics 2016 Industrial Robots", available at http://www.ifr.org/industrial-robots/statistics/

General-Reference Books

[4] R. Paul, *Robot Manipulators*, MIT Press, Cambridge, MA, 1981.

[5] M. Brady et al., *Robot Motion*, MIT Press, Cambridge, MA, 1983.

[6] W. Synder, *Industrial Robots: Computer Interfacing and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

[7] Y. Koren, *Robotics for Engineers*, McGraw-Hill, New York, 1985.

[8] H. Asada and J.J. Slotine, *Robot Analysis and Control*, Wiley, New York, 1986.

[9] K. Fu, R. Gonzalez, and C.S.G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill, New York, 1987.

[10] E. Riven, *Mechanical Design of Robots*, McGraw-Hill, New York, 1988.

[11] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.

[12] M. Spong, *Robot Control: Dynamics, Motion Planning, and Analysis*, IEEE Press, New York, 1992.

[13] S.Y. Nof, *Handbook of Industrial Robotics*, 2nd Edition, Wiley, New York, 1999.

[14] L.W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*, Wiley, New York, 1999.

[15] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*, 2nd Edition, Springer-Verlag, London, 2000.

[16] P.I. Corke, "Robotics, Vision & Control", Springer 2011, ISBN 978-3-642-20143-1.

General-Reference Journals and Magazines

[17] IEEE/ASME Transactions on Mechatronics.

[18] *IEEE Transactions on Robotics and Automation*.

[19] *International Journal of Robotics Research (MIT Press)*.

[20] *ASME Journal of Dynamic Systems, Measurement, and Control*.

[21] *International Journal of Robotics & Automation (IASTED)*.

EXERCISES

1.1 [20] Make a chronology of major events in the development of industrial robots over the past 40 years. See the Bibliography and general references.

1.2 [20] Make a chart showing the major applications of industrial robots (e.g., spot welding, assembly, etc.) and the percentage of installed robots in use in each application area. Base your chart on the most recent data you can find. See the Bibliography and general references.

1.3 [40] Figure 1.3 shows how the cost effectiveness of industrial robots is increasing. Find data on the cost of human labor in various specific industries (e.g., labor in the auto industry, labor in the electronics assembly industry, labor in agriculture, etc.) and create a graph showing how these costs compare to the use of robotics. You should see that the robot cost curve "crosses" various human cost curves of different industries at different times. From this, derive approximate dates when robotics first became cost effective for use in various industries.

1.4 [10] In a sentence or two, define kinematics, workspace, and trajectory.

1.5 [10] In a sentence or two, define frame, degree of freedom, and position control.

1.6 [10] In a sentence or two, define force control, and robot programming language.

1.7 [10] In a sentence or two, define nonlinear control, and off-line programming.

1.8 [20] Make a chart indicating how labor costs have risen over the past 20 years.

1.9 [20] Make a chart indicating how the computer performance–price ratio has increased over the past 20 years.

1.10 [20] Make a chart showing the major users of industrial robots (e.g., aerospace, automotive, etc.) and the percentage of installed robots in use in each industry. Base your chart on the most recent data you can find (see the reference section).

1.11 [30] Write a robot program using simple terms such as "move to" that will deal cards from a pre-shuffled deck to four players. Employ the following vectors to denote the player locations: P_1, P_2, P_3, and P_4. Let P_{deck} be the coordinates for the deck, where a card is presented to the manipulator. Assume that the robot can hold one card at a time in a gripper that responds to the commands "open" and "close."

1.12 [20] From your experience or exposure, give examples of both mechanical manipulators and fixed automation machines. These could be applications or specific devices.

1.13 [20] Why is six the minimum number of joints for a *universal robot*?

1.14 [15] Using the notation of this book, compute

$$^{A}P_3 = s_1\, ^{A}P_1 + c_2\, ^{A}P_2$$

$$\text{if } \theta_1 = \frac{\pi}{6}, \theta_2 = \frac{\pi}{3}, \; ^{A}P_1^{\mathrm{T}} = [3, 1, 5], \text{ and } ^{A}P_2 = \begin{bmatrix} 2 \\ 6 \\ 9 \end{bmatrix}.$$

1.15 [20] For the manipulator of Fig. 1.10 with electric motors supplying torques τ_1, τ_2, and τ_3, list pros and cons of placing the motors at the joints vs. using a belt-drive system to place the motors at the robot base.

PROGRAMMING EXERCISE (PART 1)

Familiarize yourself with the computer you will use to do the programming exercises at the end of each chapter. Make sure you can create and edit files, and can compile and execute programs.

MATLAB EXERCISE 1

At the end of most chapters in this textbook, a MATLAB exercise is given. Generally, these exercises ask the student to program the pertinent robotics mathematics in MATLAB and then check the results of the MATLAB Robotics Toolbox. The textbook assumes familiarity with MATLAB and linear algebra (matrix theory). Also, the student must become familiar with the MATLAB Robotics Toolbox. For MATLAB Exercise 1,

a) Familiarize yourself with the MATLAB programming environment if necessary. At the MATLAB software prompt, try typing *demo* and *help*. Using the color-coded MATLAB editor, learn how to create, edit, save, run, and debug m-files (ASCII files with series of MATLAB statements). Learn how to create arrays (matrices and vectors), and explore the built-in MATLAB linear-algebra functions for matrix and vector multiplication, dot and cross products, transposes, determinants, and inverses, and for the solution of linear equations. MATLAB is based on the language C, but is generally much easier to use. Learn how to program logical constructs and loops in MATLAB. Learn how to use subprograms and functions. Learn how to use comments (%) for explaining your programs and tabs for easy readability. Check out www.mathworks.com for more information and tutorials. Advanced MATLAB users should become familiar with Simulink, the graphical interface of MATLAB, and with the MATLAB Symbolic Toolbox.

b) Familiarize yourself with the MATLAB Robotics Toolbox, a third-party toolbox developed by Peter I. Corke of CSIRO, Pinjarra Hills, Australia. This product can be downloaded for free from http://petercorke.com/Robotics_Toolbox.html. Download the MATLAB Robotics Toolbox, and install it on your computer by using the *.zip* file and following the instructions. Read the *README* file, and familiarize yourself with the various functions available to the user. Find the *robot.pdf* file—this is the user manual giving background information and detailed usage of all of the Toolbox functions. Don't worry if you can't understand the purpose of these functions yet; they deal with robotics mathematics concepts covered in Chapters 2 through 9 of this book.

CHAPTER 2

Spatial Descriptions and Transformations

2.1 INTRODUCTION

Robotic manipulation, by definition, implies that parts and tools will be moved around in space by some sort of mechanism. This naturally leads to a need for representing positions and the orientations of parts, of tools, and of the mechanism itself. To define and manipulate mathematical quantities that represent position and orientation, we must define coordinate systems and develop conventions for representation. Many of the ideas developed here in the context of position and orientation will form a basis for our later consideration of linear and rotational velocities, forces, and torques.

We adopt the philosophy that somewhere there is a **universe coordinate system** to which everything we discuss can be referenced. We will describe all positions and orientations with respect to the universe coordinate system or with respect to other Cartesian coordinate systems that are (or could be) defined relative to the universe system.

2.2 DESCRIPTIONS: POSITIONS, ORIENTATIONS, AND FRAMES

A **description** is used to specify attributes of various objects with which a manipulation system deals. These objects are parts, tools, and the manipulator itself. In this section, we discuss the description of positions, of orientations, and of an entity that contains both of these descriptions: the frame.

Description of a Position

Once a coordinate system is established, we can locate any point in the universe with a 3×1 **position vector**. Because we will often define many coordinate systems in addition to the universe coordinate system, vectors must be tagged with information identifying which coordinate system within which they are defined. In this book, vectors are written with a leading superscript indicating the coordinate system to which they are referenced (unless it is clear from context)—for example, ${}^{A}P$. This means that the components of ${}^{A}P$ have numerical values that indicate distances along the axes of $\{A\}$. Each of these distances along an axis can be thought of as the result of projecting the vector onto the corresponding axis.

Figure 2.1 pictorially represents a coordinate system, $\{A\}$, with three mutually orthogonal unit vectors with solid heads. A point ${}^{A}P$ is represented as a vector and can equivalently be thought of as a position in space, or simply as an ordered set of three numbers. Individual elements of a vector are given the subscripts x, y, and z:

$$ {}^{A}P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \tag{2.1} $$

In summary, we will describe the position of a point in space with a position vector. Other 3-tuple descriptions of the position of points, such as spherical or cylindrical coordinate representations, will be discussed in the exercises at the end of the chapter.

Description of an Orientation

Often, we will find it necessary not only to represent a point in space, but also to describe the **orientation** of a body in space. For example, if vector ${}^{A}P$ in Fig. 2.2 locates the point directly between the fingertips of a manipulator's hand, the complete location of the hand is still not specified until its orientation is also given. Assuming that the manipulator has a sufficient number of joints,[1] the hand could

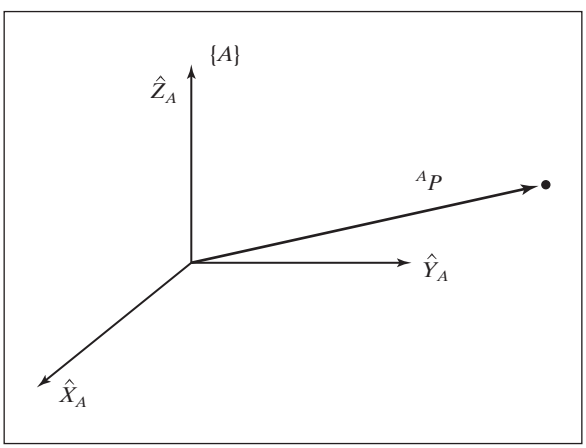

FIGURE 2.1: Vector relative to frame (example).

[1] How many are "sufficient" will be discussed in Chapters 3 and 4.

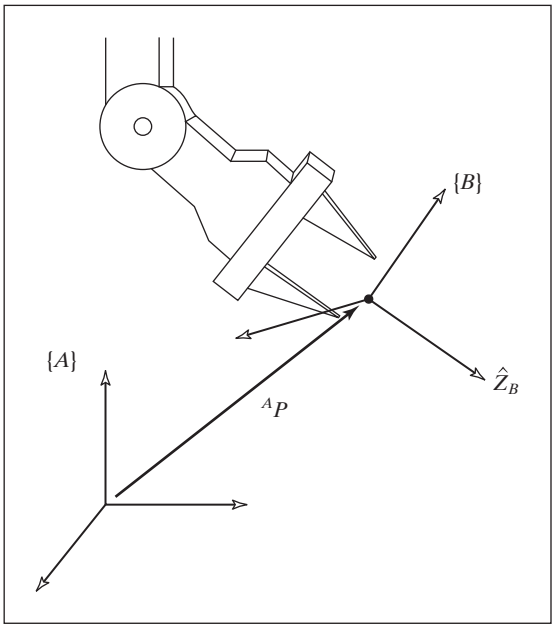

FIGURE 2.2: Locating an object in position and orientation.

be *oriented* arbitrarily while keeping the point between the fingertips at the same position in space. In order to describe the orientation of a body, we will *attach a coordinate system to the body and then give a description of this coordinate system relative to the reference system*. In Fig. 2.2, coordinate system $\{B\}$ has been attached to the body in a known way. A description of $\{B\}$ relative to $\{A\}$ now suffices to give the orientation of the body.

Thus, positions of points are described with vectors, and orientations of bodies are described with an attached coordinate system. One way to describe the body-attached coordinate system, $\{B\}$, is to write the unit vectors of its three principal axes[2] in terms of the coordinate system $\{A\}$.

We denote the unit vectors giving the principal directions of coordinate system $\{B\}$ as $\hat{X}_B$, $\hat{Y}_B$, and $\hat{Z}_B$. When written in terms of coordinate system $\{A\}$, they are called ${}^A\hat{X}_B$, ${}^A\hat{Y}_B$, and ${}^A\hat{Z}_B$. It will be convenient if we stack these three unit vectors together as the columns of a 3×3 matrix, in the order ${}^A\hat{X}_B$, ${}^A\hat{Y}_B$, ${}^A\hat{Z}_B$. We will call this matrix a **rotation matrix**, and, because this particular rotation matrix describes $\{B\}$ relative to $\{A\}$, we name it with the notation ${}^A_B R$ (the choice of leading sub- and superscripts in the definition of rotation matrices will become clear in following sections):

$$ {}^A_B R = \begin{bmatrix} {}^A\hat{X}_B & {}^A\hat{Y}_B & {}^A\hat{Z}_B \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{2.2} $$

[2] It is often convenient to use three, although any two would suffice. (The third can always be recovered by taking the cross product of the two given.)

In summary, a set of three vectors may be used to specify an orientation. For convenience, we will construct a 3×3 matrix that has these three vectors as its columns. Hence, whereas the position of a point is represented with a vector, the orientation of a body is represented with a matrix. In Section 2.8, we will consider some other descriptions of orientation that require only three parameters.

We can give expressions for the scalars r_{ij} in (2.2) by noting that the components of any vector are simply the projections of that vector onto the unit directions of its reference frame. Hence, each component of ${}^A_B R$ in (2.2) can be written as the dot product of a pair of unit vectors:

$$ {}^A_B R = \begin{bmatrix} {}^A\hat{X}_B & {}^A\hat{Y}_B & {}^A\hat{Z}_B \end{bmatrix} = \begin{bmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{bmatrix}. \tag{2.3} $$

In this case, we have omitted the leading superscripts in the rightmost matrix of (2.3). In fact, the choice of frame in which to describe the unit vectors is arbitrary as long as it is the same for each pair being dotted. The dot product of two unit vectors yields the cosine of the angle between them, so it is clear why the components of rotation matrices are often referred to as **direction cosines**.

Further inspection of (2.3) shows that the rows of the matrix are the unit vectors of $\{A\}$ expressed in $\{B\}$; that is,

$$ {}^A_B R = \begin{bmatrix} {}^A\hat{X}_B & {}^A\hat{Y}_B & {}^A\hat{Z}_B \end{bmatrix} = \begin{bmatrix} {}^B\hat{X}_A^T \\ {}^B\hat{Y}_A^T \\ {}^B\hat{Z}_A^T \end{bmatrix}. \tag{2.4} $$

Hence, ${}^B_A R$, the description of frame $\{A\}$ relative to $\{B\}$, is given by the transpose of (2.3); that is,

$$ {}^B_A R = {}^A_B R^T. \tag{2.5} $$

This suggests that the inverse of a rotation matrix is equal to its transpose, a fact that can be easily verified as

$$ {}^A_B R^T \; {}^A_B R = \begin{bmatrix} {}^A\hat{X}_B^T \\ {}^A\hat{Y}_B^T \\ {}^A\hat{Z}_B^T \end{bmatrix} \begin{bmatrix} {}^A\hat{X}_B & {}^A\hat{Y}_B & {}^A\hat{Z}_B \end{bmatrix} = I_3, \tag{2.6} $$

where I_3 is the 3×3 identity matrix. Hence,

$$ {}^A_B R = {}^B_A R^{-1} = {}^B_A R^T. \tag{2.7} $$

Indeed, from linear algebra [1], we know that the inverse of a matrix with orthonormal columns is equal to its transpose. We have just shown this geometrically.

Description of a Frame

The information needed to completely specify the whereabouts of the manipulator hand in Fig. 2.2 is a position and an orientation. The point on the body whose position

we describe could be chosen arbitrarily, however. *For convenience, the point whose position we will describe is chosen as the origin of the body-attached frame*. The situation of a position and an orientation pair arises so often in robotics that we define an entity called a **frame**, which is a set of four vectors giving position and orientation information. For example, in Fig. 2.2, one vector locates the fingertip position, and three more describe its orientation. Equivalently, the description of a frame can be thought of as a position vector and a rotation matrix. Note that a frame is a coordinate system where, in addition to the orientation, we give a position vector which locates its origin relative to some other embedding frame. For example, frame $\{B\}$ is described by ${}^{A}_{B}R$ and ${}^{A}P_{BORG}$, where ${}^{A}P_{BORG}$ is the vector that locates the origin of the frame $\{B\}$:

$$\{B\} = \{{}^{A}_{B}R,\ {}^{A}P_{BORG}\}. \tag{2.8}$$

In Fig. 2.3, there are three frames that are shown along with the universe coordinate system. Frames $\{A\}$ and $\{B\}$ are known relative to the universe coordinate system, and frame $\{C\}$ is known relative to frame $\{A\}$.

In Fig. 2.3, we introduce a *graphical representation* of frames, which is convenient in visualizing frames. A frame is depicted by three arrows representing unit vectors defining the principal axes of the frame. An arrow representing a vector is drawn from one origin to another. This vector represents the position of the origin at the head of the arrow in terms of the frame at the tail of the arrow. The direction of this locating arrow tells us, for example, in Fig. 2.3, that $\{C\}$ is known relative to $\{A\}$, and not vice versa.

In summary, a frame can be used as a description of one coordinate system relative to another. A frame encompasses two ideas by representing both position and orientation, and so may be thought of as a generalization of those two ideas. Positions could be represented by a frame whose rotation-matrix part is the identity matrix and whose position-vector part locates the point being described. Likewise,

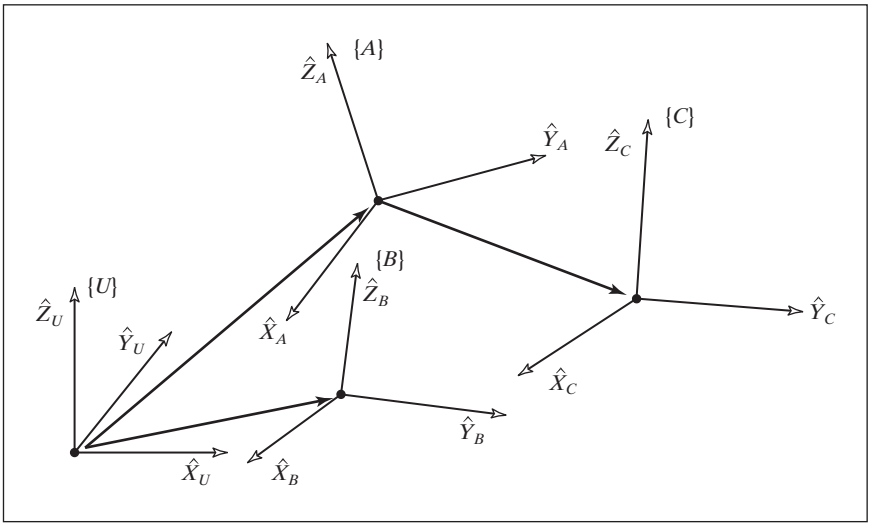

FIGURE 2.3: Example of several frames.

an orientation could be represented by a frame whose position-vector part was the zero vector.

2.3 MAPPINGS: CHANGING DESCRIPTIONS FROM FRAME TO FRAME

In a great many of the problems in robotics, we are concerned with expressing the same quantity in terms of various reference coordinate systems. The previous section introduced descriptions of positions, orientations, and frames; we now consider the mathematics of **mapping** in order to change descriptions from frame to frame.

Mappings Involving Translated Frames

In Fig. 2.4, we have a position defined by the vector ${}^{B}P$. We wish to express this point in space in terms of frame $\{A\}$, when $\{A\}$ has the same orientation as $\{B\}$. In this case, $\{B\}$ differs from $\{A\}$ only by a *translation*, which is given by ${}^{A}P_{BORG}$, a vector that locates the origin of $\{B\}$ relative to $\{A\}$.

Because both vectors are defined relative to frames of the same orientation, we calculate the description of point P relative to $\{A\}$, ${}^{A}P$, by vector addition:

$$ {}^{A}P = {}^{B}P + {}^{A}P_{BORG}. \tag{2.9}$$

Note that only in the special case of equivalent orientations may we add vectors that are defined in terms of different frames.

In this simple example, we have illustrated **mapping** a vector from one frame to another. This idea of mapping, or changing the description from one frame to another, is an extremely important concept. The quantity itself (here, a point in space) is not changed; only its description is changed. This is illustrated in Fig. 2.4,

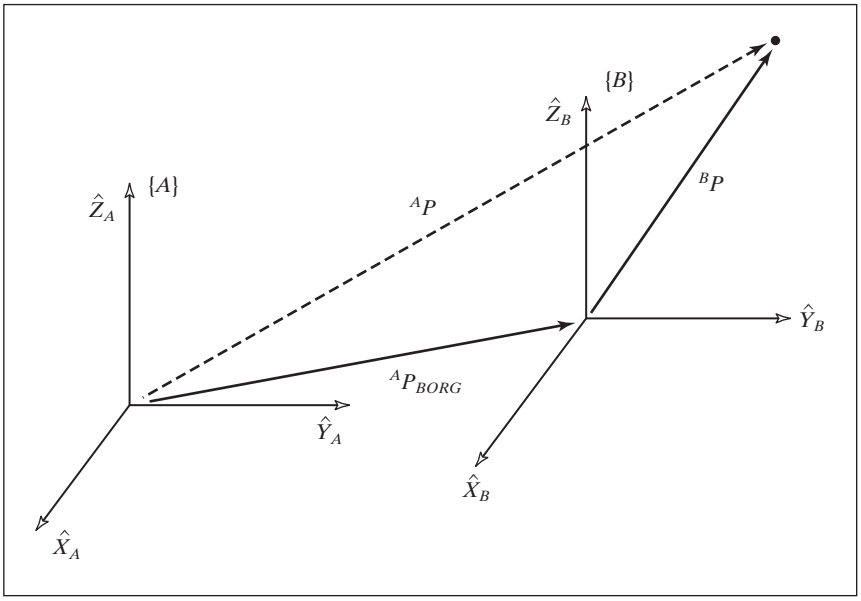

FIGURE 2.4: Translational mapping.

where the point described by ${}^{B}P$ is not translated, but remains the same, and instead we have computed a new description of the same point, but now with respect to system $\{A\}$.

We say that the vector ${}^{A}P_{BORG}$ defines this mapping because all the information needed to perform the change in description is contained in ${}^{A}P_{BORG}$ (along with the knowledge that the frames had equivalent orientation).

Mappings Involving Rotated Frames

Section 2.2 introduced the notion of describing an orientation by three unit vectors denoting the principal axes of a body-attached coordinate system. For convenience, we stack these three unit vectors together as the columns of a 3×3 matrix. We will call this matrix a rotation matrix, and, if this particular rotation matrix describes $\{B\}$ relative to $\{A\}$, we name it with the notation ${}^{A}_{B}R$.

Note that, by our definition, the columns of a rotation matrix all have unit magnitude, and, further, that these unit vectors are orthogonal. As we saw earlier, a consequence of this is that

$$ {}^{A}_{B}R = {}^{B}_{A}R^{-1} = {}^{B}_{A}R^{T}. \tag{2.10} $$

Therefore, because the columns of ${}^{A}_{B}R$ are the unit vectors of $\{B\}$ written in $\{A\}$, the *rows* of ${}^{A}_{B}R$ are the unit vectors of $\{A\}$ written in $\{B\}$.

So, a rotation matrix can be interpreted as a set of three column vectors, or as a set of three row vectors, as follows:

$$ {}^{A}_{B}R = \begin{bmatrix} {}^{A}\hat{X}_{B} & {}^{A}\hat{Y}_{B} & {}^{A}\hat{Z}_{B} \end{bmatrix} = \begin{bmatrix} {}^{B}\hat{X}_{A}^{T} \\ {}^{B}\hat{Y}_{A}^{T} \\ {}^{B}\hat{Z}_{A}^{T} \end{bmatrix}. \tag{2.11} $$

As in Fig. 2.5, the situation will arise often where we know the definition of a vector with respect to some frame, $\{B\}$, and we would like to know its definition with respect to another frame, $\{A\}$, where the origins of the two frames are coincident. This computation is possible when a description of the orientation of $\{B\}$ is known relative to $\{A\}$. This orientation is given by the rotation matrix ${}^{A}_{B}R$, whose columns are the unit vectors of $\{B\}$ written in $\{A\}$.

In order to calculate ${}^{A}P$, we note that the components of any vector are simply the projections of that vector onto the unit directions of its frame. The projection is calculated as the vector dot product. Thus, we see that the components of ${}^{A}P$ may be calculated as

$$ \begin{aligned} {}^{A}p_{x} &= {}^{B}\hat{X}_{A} \cdot {}^{B}P, \\ {}^{A}p_{y} &= {}^{B}\hat{Y}_{A} \cdot {}^{B}P, \text{ and} \\ {}^{A}p_{z} &= {}^{B}\hat{Z}_{A} \cdot {}^{B}P. \end{aligned} \tag{2.12} $$

In order to express (2.12) in terms of a rotation matrix multiplication, we note from (2.11) that the *rows* of ${}^{A}_{B}R$ are ${}^{B}\hat{X}_{A}$, ${}^{B}\hat{Y}_{A}$, and ${}^{B}\hat{Z}_{A}$. So (2.12) may be written compactly, by using a rotation matrix, as

$$ {}^{A}P = {}^{A}_{B}R \; {}^{B}P. \tag{2.13} $$

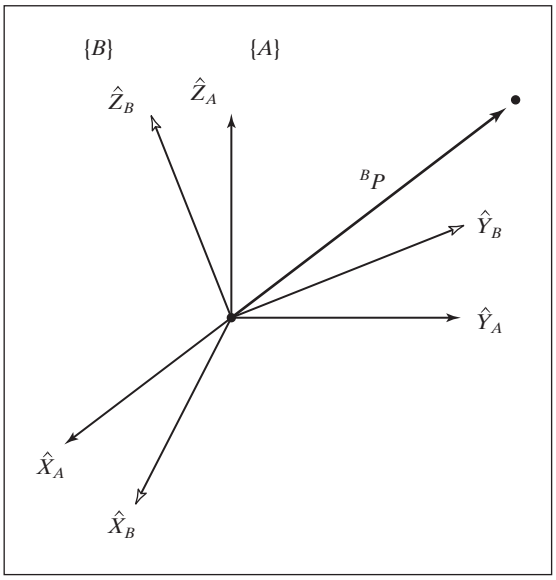

FIGURE 2.5: Rotating the description of a vector.

Equation 2.13 implements a mapping—that is, it changes the description of a vector—from ${}^BP$, which describes a point in space relative to $\{B\}$, into ${}^AP$, which is a description of the same point, but expressed relative to $\{A\}$.

We now see that our notation is of great help in keeping track of mappings and frames of reference. A helpful way of viewing the notation we have introduced is to imagine that leading subscripts cancel the leading superscripts of the following entity, for example, the Bs in (2.13).

EXAMPLE 2.1

Figure 2.6 shows a frame $\{B\}$ that is rotated relative to frame $\{A\}$ about $\hat{Z}$ by 30 degrees. Here, $\hat{Z}$ is pointing out of the page.

Writing the unit vectors of $\{B\}$ in terms of $\{A\}$ and stacking them as the columns of the rotation matrix, we obtain

$$ {}^A_BR = \begin{bmatrix} 0.866 & -0.500 & 0.000 \\ 0.500 & 0.866 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}. \tag{2.14} $$

Given

$$ {}^BP = \begin{bmatrix} 0.0 \\ 2.0 \\ 0.0 \end{bmatrix}, \tag{2.15} $$

we calculate ${}^AP$ as

$$ {}^AP = {}^A_BR\ {}^BP = \begin{bmatrix} -1.000 \\ 1.732 \\ 0.000 \end{bmatrix}. \tag{2.16} $$

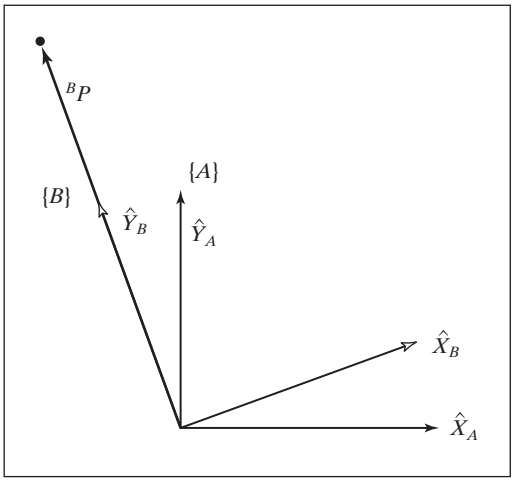

FIGURE 2.6: $\{B\}$ rotated 30 degrees about $\hat{Z}$.

Here, ${}^A_BR$ acts as a mapping that is used to describe ${}^BP$ relative to frame $\{A\}$, ${}^AP$. As was introduced in the case of translations, it is important to remember that, viewed as a mapping, the original vector P is not changed in space. Rather, we compute a new description of the vector relative to another frame.

Mappings Involving General Frames

Very often, we know the description of a vector with respect to some frame $\{B\}$, and we would like to know its description with respect to another frame, $\{A\}$. We now consider the general case of mapping. Here, the origin of frame $\{B\}$ is not coincident with that of frame $\{A\}$ but has a general vector offset. The vector that locates $\{B\}$'s origin is called ${}^AP_{BORG}$. Also, $\{B\}$ is rotated with respect to $\{A\}$, as described by ${}^A_BR$. Given ${}^BP$, we wish to compute ${}^AP$, as in Fig. 2.7.

We can first change ${}^BP$ to its description relative to an intermediate frame that has the same orientation as $\{A\}$, but whose origin is coincident with the origin of $\{B\}$. This is done by premultiplying by ${}^A_BR$ as in the last section. We then account for the translation between origins by simple vector addition, as before, and obtain

$$^AP = {}^A_BR\ {}^BP + {}^AP_{BORG}. \tag{2.17}$$

Equation 2.17 describes a general transformation mapping of a vector from its description in one frame to a description in a second frame. Note the following interpretation of our notation as exemplified in (2.17): the B's cancel, leaving all quantities as vectors written in terms of A, which may then be added.

The form of (2.17) is not as appealing as the conceptual form

$$^AP = {}^A_BT\ {}^BP. \tag{2.18}$$

That is, we would like to think of a mapping from one frame to another as an operator in matrix form. This aids in writing compact equations, and is conceptually

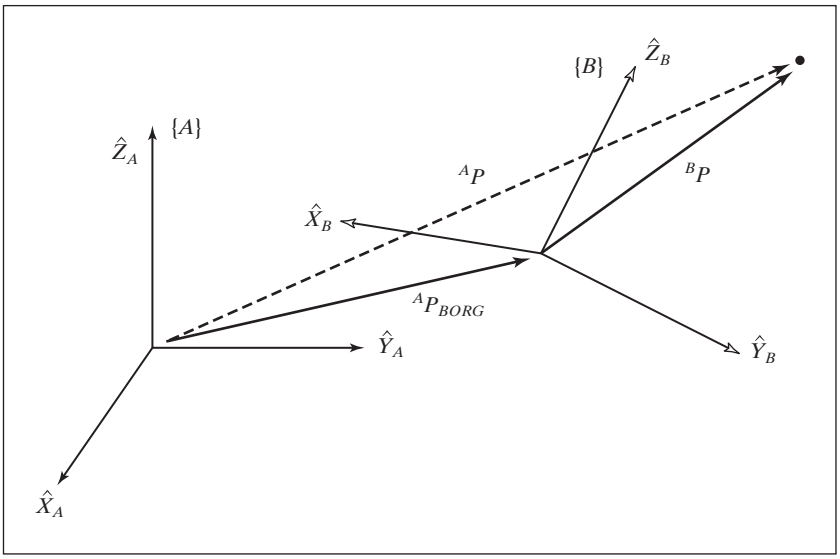

FIGURE 2.7: General transform of a vector.

clearer than (2.17). In order that we may write the mathematics given in (2.17) in the matrix operator form suggested by (2.18), we define a 4×4 matrix operator and use 4×1 position vectors, so that (2.18) has the structure

$$\begin{bmatrix} {}^AP \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & {}^A_BR & & {}^AP_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^BP \\ 1 \end{bmatrix}. \tag{2.19}$$

In other words,

1. a "1" is added as the last element of the 4×1 vectors, and;
2. a row "[0 0 0 1]" is added as the last row of the 4×4 matrix.

We adopt the convention that a position vector is 3×1 or 4×1, depending on whether it appears multiplied by a 3×3 matrix or by a 4×4 matrix. It is readily seen that (2.19) implements

$$\begin{aligned} {}^AP &= {}^A_BR\ {}^BP + {}^AP_{BORG} \\ 1 &= 1. \end{aligned} \tag{2.20}$$

The 4×4 matrix in (2.19) is called a **homogeneous transform**. For our purposes, it can be regarded purely as a construction used to cast the rotation and translation of the general transform into a single matrix form. In other fields of study, it can be used to compute perspective and scaling operations (when the last row is other than "[0 0 0 1]" or the rotation matrix is not orthonormal). The interested reader should see [2].

Often, we will write an equation like (2.18) without any notation indicating that it is a homogeneous representation, because it is obvious from context. Note

that, although homogeneous transforms are useful in writing compact equations, a computer program to transform vectors would generally not use them, because of time wasted multiplying ones and zeros. Thus, this representation is mainly for our convenience when thinking and writing equations down on paper.

Just as we used rotation matrices to specify an orientation, we will use transforms (usually in homogeneous representation) to specify a frame. Observe that, although we have introduced homogeneous transforms in the context of mappings, they also serve as descriptions of frames. The description of frame $\{B\}$ relative to $\{A\}$ is ${}^{A}_{B}T$.

EXAMPLE 2.2

Figure 2.8 shows a frame $\{B\}$, which is rotated relative to frame $\{A\}$ about $\hat{Z}_A$ by 30 degrees, translated 10 units in $\hat{X}_A$, and translated 5 units in $\hat{Y}_A$. Find ${}^{A}P$, where ${}^{B}P = [3.0 \;\; 7.0 \;\; 0.0]^T$.

The definition of frame $\{B\}$ is

$$ {}^{A}_{B}T = \begin{bmatrix} 0.866 & -0.500 & 0.000 & 10.0 \\ 0.500 & 0.866 & 0.000 & 5.0 \\ 0.000 & 0.000 & 1.000 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.21} $$

Given

$$ {}^{B}P = \begin{bmatrix} 3.0 \\ 7.0 \\ 0.0 \end{bmatrix}, \tag{2.22} $$

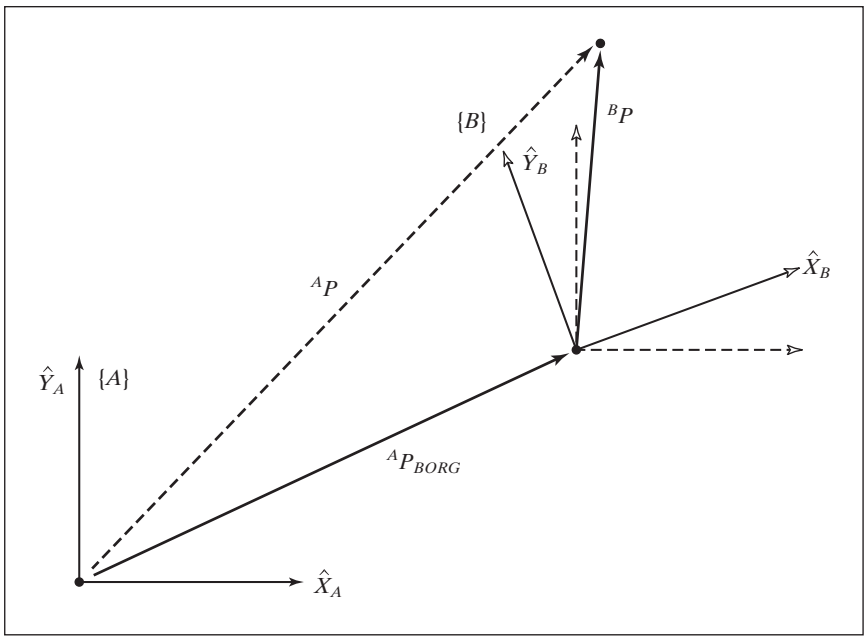

FIGURE 2.8: Frame $\{B\}$ rotated and translated.

we use the definition of $\{B\}$ just given as a transformation:

$$^{A}P = {}^{A}_{B}T\ {}^{B}P = \begin{bmatrix} 9.098 \\ 12.562 \\ 0.000 \end{bmatrix}. \tag{2.23}$$

2.4 OPERATORS: TRANSLATIONS, ROTATIONS, AND TRANSFORMATIONS

The same mathematical forms used to map points between frames can also be interpreted as operators that translate points, rotate vectors, or do both. This section illustrates this interpretation of the mathematics we have already developed.

Translational Operators

A translation moves a point in space a finite distance along a given vector direction. With this interpretation of actually translating the point in space, only one coordinate system need be involved. It turns out that translating the point in space is accomplished with the same mathematics as mapping the point to a second frame. Almost always, it is very important to understand which interpretation of the mathematics is being used. The distinction is as simple as this: When a vector is moved "forward" relative to a frame, we may consider either that the vector moved "forward" or that the frame moved "backward." The mathematics involved in the two cases is identical; only our view of the situation is different. Figure 2.9 indicates

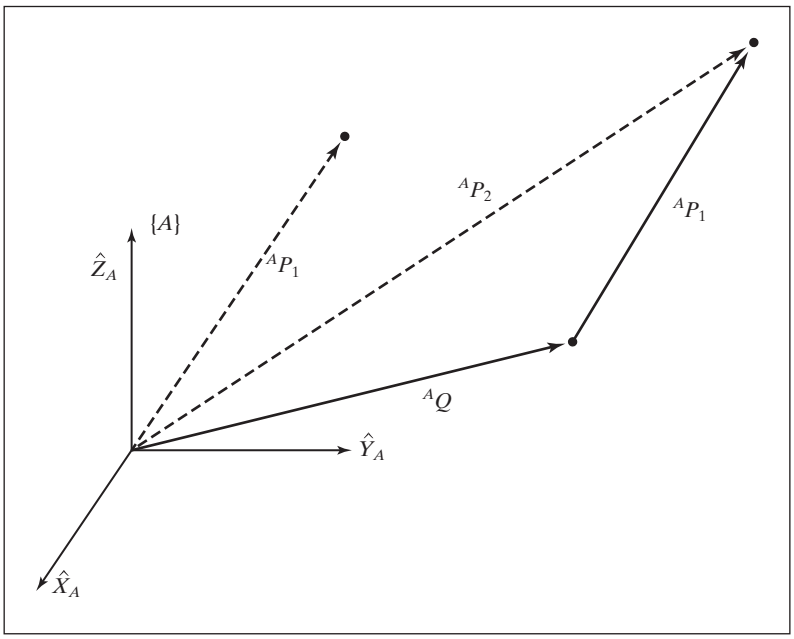

FIGURE 2.9: Translation operator.

pictorially how a vector ${}^{A}P_1$ is translated by a vector ${}^{A}Q$. Here, the vector ${}^{A}Q$ gives the information needed to perform the translation.

The result of the operation is a new vector ${}^{A}P_2$, calculated as

$${}^{A}P_2 = {}^{A}P_1 + {}^{A}Q. \tag{2.24}$$

To write this translation operation as a matrix operator, we use the notation

$${}^{A}P_2 = D_Q(q)\ {}^{A}P_1, \tag{2.25}$$

where q is the signed magnitude of the translation along the vector direction $\hat{Q}$. The D_Q operator may be thought of as a homogeneous transform of a special simple form:

$$D_Q(q) = \begin{bmatrix} 1 & 0 & 0 & q_x \\ 0 & 1 & 0 & q_y \\ 0 & 0 & 1 & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.26}$$

where q_x, q_y, and q_z are the components of the translation vector Q and $q = \sqrt{q_x^2 + q_y^2 + q_z^2}$. Equations (2.9) and (2.24) implement the same mathematics. Note that, if we had defined ${}^{B}P_{AORG}$ (instead of ${}^{A}P_{BORG}$) in Fig. 2.4 and had used it in (2.9), then we would have seen a sign change between (2.9) and (2.24). This sign change would indicate the difference between moving the vector "forward" and moving the coordinate system "backward." By defining the location of $\{B\}$ relative to $\{A\}$ (with ${}^{A}P_{BORG}$), we cause the mathematics of the two interpretations to be the same. Now that the "D_Q" notation has been introduced, we may also use it to describe frames and as a mapping.

Rotational Operators

Another interpretation of a rotation matrix is as a *rotational operator* that operates on a vector ${}^{A}P_1$ and changes that vector to a new vector, ${}^{A}P_2$, by means of a rotation, R. Usually, when a rotation matrix is shown as an operator, no sub- or superscripts appear, because it is not viewed as relating two frames. That is, we may write

$${}^{A}P_2 = R\ {}^{A}P_1. \tag{2.27}$$

Again, as in the case of translations, the mathematics described in (2.13) and in (2.27) are the same; only our interpretation is different. This fact also allows us to see *how to obtain* rotational matrices that are to be used as operators:

The rotation matrix that rotates vectors through some rotation, R, is the same as the rotation matrix that describes a frame rotated by R relative to the reference frame.

Although a rotation matrix is easily viewed as an operator, we will also define another notation for a rotational operator that clearly indicates which axis is being rotated about:

$${}^{A}P_2 = R_K(\theta)\ {}^{A}P_1. \tag{2.28}$$

In this notation, "$R_K(\theta)$" is a rotational operator that performs a rotation about the axis direction $\hat{K}$ by θ degrees. This operator can be written as a homogeneous

transform whose position-vector part is zero. For example, substitution into (2.11) yields the operator that rotates about the $\hat{Z}$ axis by θ as

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.29}$$

Of course, to rotate a position vector, we could just as well use the 3×3 rotation-matrix part of the homogeneous transform. The "R_K" notation, therefore, may be considered to represent a 3×3 or a 4×4 matrix. Later in this chapter, we will see how to write the rotation matrix for a rotation about a general axis $\hat{K}$.

EXAMPLE 2.3

Figure 2.10 shows a vector ${}^A P_1$. We wish to compute the vector obtained by rotating this vector about $\hat{Z}$ by 30 degrees. We will call the new vector ${}^A P_2$.

The rotation matrix that rotates vectors by 30 degrees about $\hat{Z}$ is the same as the rotation matrix that describes a frame rotated 30 degrees about $\hat{Z}$ relative to the reference frame. Thus, the correct rotational operator is

$$R_z(30.0) = \begin{bmatrix} 0.866 & -0.500 & 0.000 \\ 0.500 & 0.866 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}. \tag{2.30}$$

Given

$${}^A P_1 = \begin{bmatrix} 0.0 \\ 2.0 \\ 0.0 \end{bmatrix}, \tag{2.31}$$

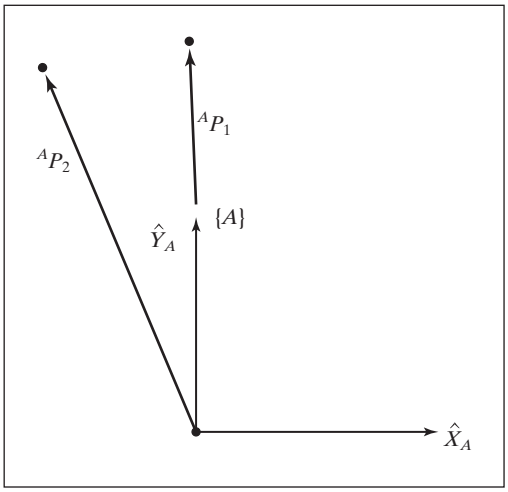

FIGURE 2.10: The vector ${}^A P_1$ rotated 30 degrees about $\hat{Z}$.

we calculate ${}^{A}P_2$ as

$$ {}^{A}P_2 = R_z(30.0)\ {}^{A}P_1 = \begin{bmatrix} -1.000 \\ 1.732 \\ 0.000 \end{bmatrix}. \tag{2.32} $$

Equations (2.13) and (2.27) implement the same mathematics. Note that, if we had defined ${}^{B}_{A}R$ (instead of ${}^{A}_{B}R$) in (2.13), then the inverse of R would appear in (2.27). This change would indicate the difference between rotating the vector "forward" versus rotating the coordinate system "backward." By defining the location of $\{B\}$ relative to $\{A\}$ (by ${}^{A}_{B}R$), we cause the mathematics of the two interpretations to be the same.

Transformation Operators

As with vectors and rotation matrices, a frame has another interpretation as a *transformation operator*. In this interpretation, only one coordinate system is involved, so the symbol T is used without sub- or superscripts. The operator T rotates and translates a vector ${}^{A}P_1$ to compute a new vector,

$$ {}^{A}P_2 = T\ {}^{A}P_1. \tag{2.33} $$

Again, as in the case of rotations, the mathematics described in (2.18) and in (2.33) are the same, only our interpretation is different. This fact also allows us to see how to obtain homogeneous transforms that are to be used as operators:

The transform that rotates by R and translates by Q is the same as the transform that describes a frame rotated by R and translated by Q relative to the reference frame.

A transform is usually thought of as being in the form of a homogeneous transform with general rotation-matrix and position-vector parts.

EXAMPLE 2.4

Figure 2.11 shows a vector ${}^{A}P_1$. We wish to rotate it about $\hat{Z}_A$ by 30 degrees and translate it 10 units in $\hat{X}_A$ and 5 units in $\hat{Y}_A$. Find ${}^{A}P_2$, where ${}^{A}P_1 = [3.0\ 7.0\ 0.0]^T$.

The operator T, which performs the translation and rotation, is

$$ T = \begin{bmatrix} 0.866 & -0.500 & 0.000 & 10.0 \\ 0.500 & 0.866 & 0.000 & 5.0 \\ 0.000 & 0.000 & 1.000 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.34} $$

Given

$$ {}^{A}P_1 = \begin{bmatrix} 3.0 \\ 7.0 \\ 0.0 \end{bmatrix}, \tag{2.35} $$

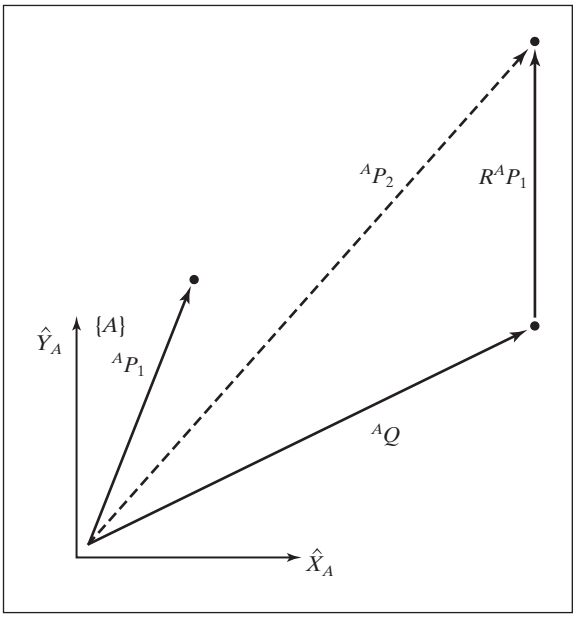

FIGURE 2.11: The vector ${}^{A}P_1$ rotated and translated to form ${}^{A}P_2$.

we use T as an operator:

$$ {}^{A}P_2 = T\ {}^{A}P_1 = \begin{bmatrix} 9.098 \\ 12.562 \\ 0.000 \end{bmatrix}. \tag{2.36} $$

Note this example is numerically exactly the same as Example 2.2, but the interpretation is quite different.

2.5 SUMMARY OF INTERPRETATIONS

We have introduced concepts first for the case of translation only, then for the case of rotation only, and finally for the general case of rotation about a point and translation of that point. Having understood the general case of rotation and translation, we will not need to explicitly consider the two simpler cases, since they are contained within the general framework.

As a general tool to represent frames, we have introduced the *homogeneous transform*, a 4×4 matrix containing orientation and position information.

We have introduced three interpretations of this homogeneous transform:

1. It is a *description of a frame*. ${}^{A}_{B}T$ describes the frame $\{B\}$ relative to the frame $\{A\}$. Specifically, the columns of ${}^{A}_{B}R$ are unit vectors defining the directions of the principal axes of $\{B\}$, and ${}^{A}P_{BORG}$ locates the position of the origin of $\{B\}$.
2. It is a *transform mapping*. ${}^{A}_{B}T$ maps ${}^{B}P \rightarrow {}^{A}P$.
3. It is a *transform operator*. T operates on ${}^{A}P_1$ to create ${}^{A}P_2$.

From this point on, the terms *frame* and *transform* will both be used to refer to a position vector plus an orientation. *Frame* is the term favored in speaking of a description, and *transform* is used most frequently when function as a mapping or operator is implied. Note that transformations are generalizations of (and subsume) translations and rotations; we will often use the term *transform* when speaking of a pure rotation (or translation).

2.6 TRANSFORMATION ARITHMETIC

In this section, we look at the multiplication of transforms and the inversion of transforms. These two elementary operations form a functionally complete set of transform operators.

Compound Transformations

In Fig. 2.12, we have ${}^{C}P$ and wish to find ${}^{A}P$.

Frame $\{C\}$ is known relative to frame $\{B\}$, and frame $\{B\}$ is known relative to frame $\{A\}$. We can transform ${}^{C}P$ into ${}^{B}P$ as

$$ {}^{B}P = {}^{B}_{C}T \; {}^{C}P; \tag{2.37} $$

then, we can transform ${}^{B}P$ into ${}^{A}P$ as

$$ {}^{A}P = {}^{A}_{B}T \; {}^{B}P. \tag{2.38} $$

Combining (2.37) and (2.38), we get the (not unexpected) result

$$ {}^{A}P = {}^{A}_{B}T \; {}^{B}_{C}T \; {}^{C}P, \tag{2.39} $$

from which we could define

$$ {}^{A}_{C}T = {}^{A}_{B}T \; {}^{B}_{C}T. \tag{2.40} $$

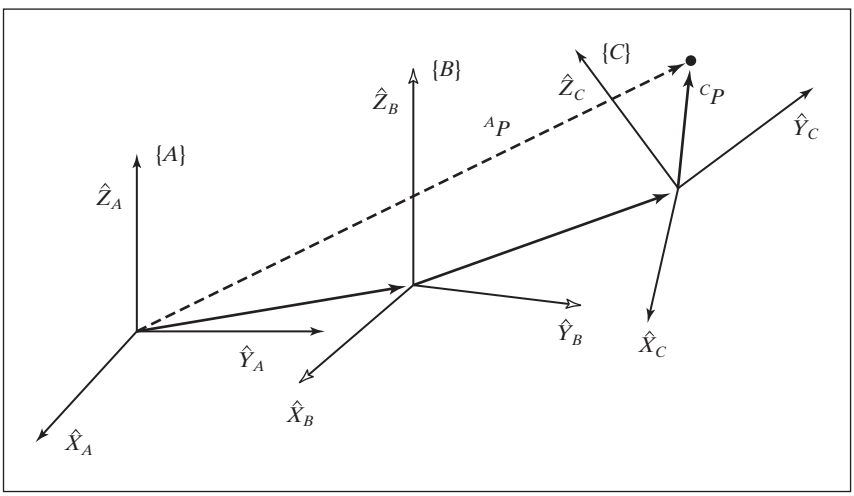

FIGURE 2.12: Compound frames: each is known relative to the previous one.

Again, note that familiarity with the sub- and superscript notation makes these manipulations simple. In terms of the known descriptions of $\{B\}$ and $\{C\}$, we can give the expression for ${}^{A}_{C}T$ as

$$ {}^{A}_{C}T = \left[\begin{array}{cccc|c} & {}^{A}_{B}R & & {}^{B}_{C}R & {}^{A}_{B}R\ {}^{B}P_{CORG} + {}^{A}P_{BORG} \\ \hline 0 & 0 & 0 & & 1 \end{array}\right]. \tag{2.41} $$

Inverting a Transform

Consider a frame $\{B\}$ that is known with respect to a frame $\{A\}$—that is, we know the value of ${}^{A}_{B}T$. Sometimes we will wish to invert this transform, in order to get a description of $\{A\}$ relative to $\{B\}$—that is, ${}^{B}_{A}T$. A straightforward way of calculating the inverse is to compute the inverse of the 4×4 homogeneous transform. However, if we do so, we are not taking full advantage of the structure inherent in the transform. It is easy to find a computationally simpler method of computing the inverse, one that does take advantage of this structure.

To find ${}^{B}_{A}T$, we must compute ${}^{B}_{A}R$ and ${}^{B}P_{AORG}$ from ${}^{A}_{B}R$ and ${}^{A}P_{BORG}$. First, recall from our discussion of rotation matrices that

$$ {}^{B}_{A}R = {}^{A}_{B}R^{T}. \tag{2.42} $$

Next, we change the description of ${}^{A}P_{BORG}$ into $\{B\}$ by using (2.13):

$$ {}^{B}({}^{A}P_{BORG}) = {}^{B}_{A}R\ {}^{A}P_{BORG} + {}^{B}P_{AORG}. \tag{2.43} $$

The left-hand side of (2.43) must be zero, so we have

$$ {}^{B}P_{AORG} = -{}^{B}_{A}R\ {}^{A}P_{BORG} = -{}^{A}_{B}R^{T}\,{}^{A}P_{BORG}. \tag{2.44} $$

Using (2.42) and (2.44), we can write the form of ${}^{B}_{A}T$ as

$$ {}^{B}_{A}T = \left[\begin{array}{ccc|c} & {}^{A}_{B}R^{T} & & -{}^{A}_{B}R^{T}\,{}^{A}P_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]. \tag{2.45} $$

Note that, with our notation,

$$ {}^{B}_{A}T = {}^{A}_{B}T^{-1}. $$

Equation (2.45) is a general and extremely useful way of computing the inverse of a homogeneous transform.

EXAMPLE 2.5

Figure 2.13 shows a frame $\{B\}$ that is rotated relative to frame $\{A\}$ about $\hat{Z}_A$ by 30 degrees and translated four units in $\hat{X}_A$ and three units in $\hat{Y}_A$. Thus, we have a description of ${}^{A}_{B}T$. Find ${}^{B}_{A}T$.

The frame defining $\{B\}$ is

$$ {}^{A}_{B}T = \begin{bmatrix} 0.866 & -0.500 & 0.000 & 4.0 \\ 0.500 & 0.866 & 0.000 & 3.0 \\ 0.000 & 0.000 & 1.000 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.46} $$

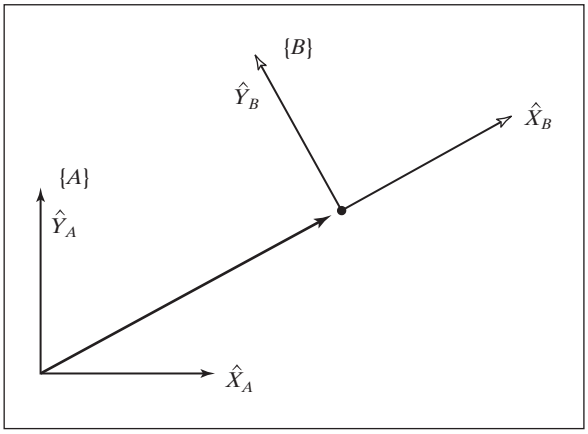

FIGURE 2.13: $\{B\}$ relative to $\{A\}$.

Using (2.45), we compute

$$ {}^{B}_{A}T = \begin{bmatrix} 0.866 & 0.500 & 0.000 & -4.964 \\ -0.500 & 0.866 & 0.000 & -0.598 \\ 0.000 & 0.000 & 1.000 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.47} $$

2.7 TRANSFORM EQUATIONS

Figure 2.14 indicates a situation in which a frame $\{D\}$ can be expressed as products of transformations in two different ways. First,

$$ {}^{U}_{D}T = {}^{U}_{A}T \; {}^{A}_{D}T; \tag{2.48} $$

second;

$$ {}^{U}_{D}T = {}^{U}_{B}T \; {}^{B}_{C}T \; {}^{C}_{D}T. \tag{2.49} $$

We can set these two descriptions of ${}^{U}_{D}T$ equal to construct a **transform equation**:

$$ {}^{U}_{A}T \; {}^{A}_{D}T = {}^{U}_{B}T \; {}^{B}_{C}T \; {}^{C}_{D}T. \tag{2.50} $$

Transform equations can be used to solve for transforms in the case of n unknown transforms and n transform equations. Consider (2.50) in the case that all transforms are known except ${}^{B}_{C}T$. Here, we have one transform equation and one unknown transform; hence, we easily find its solution to be

$$ {}^{B}_{C}T = {}^{U}_{B}T^{-1} \; {}^{U}_{A}T \; {}^{A}_{D}T \; {}^{C}_{D}T^{-1}. \tag{2.51} $$

Figure 2.15 indicates a similar situation.

Note that, in all figures, we have introduced a *graphical* representation of frames as an arrow pointing from one origin to another origin. The arrow's direction

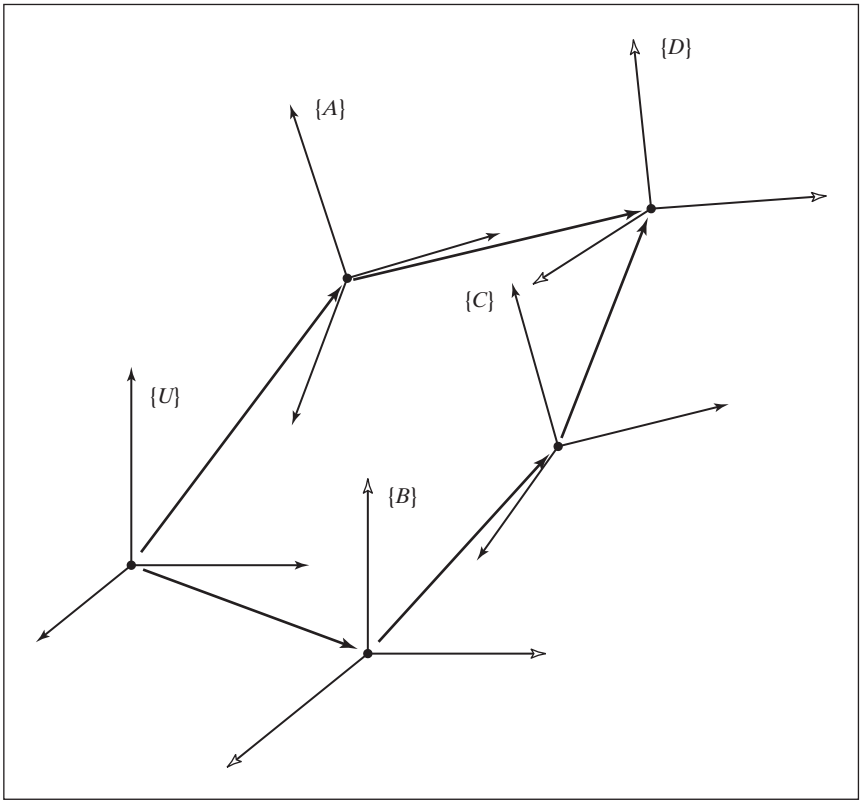

FIGURE 2.14: Set of transforms forming a loop.

indicates which way the frames are defined: In Fig. 2.14, frame $\{D\}$ is defined relative to $\{A\}$; in Fig. 2.15, frame $\{A\}$ is defined relative to $\{D\}$. In order to compound frames when the arrows line up, we simply compute the product of the transforms. If an arrow points the opposite way in a chain of transforms, we simply compute its inverse first. In Fig. 2.15, two possible descriptions of $\{C\}$ are

$$ {}^{U}_{C}T = {}^{U}_{A}T \; {}^{D}_{A}T^{-1} \; {}^{D}_{C}T \tag{2.52} $$

and

$$ {}^{U}_{C}T = {}^{U}_{B}T \; {}^{B}_{C}T. \tag{2.53} $$

Again, we might equate (2.52) and (2.53) to solve for, say, ${}^{U}_{A}T$:

$$ {}^{U}_{A}T = {}^{U}_{B}T \; {}^{B}_{C}T \; {}^{D}_{C}T^{-1} \; {}^{D}_{A}T. \tag{2.54} $$

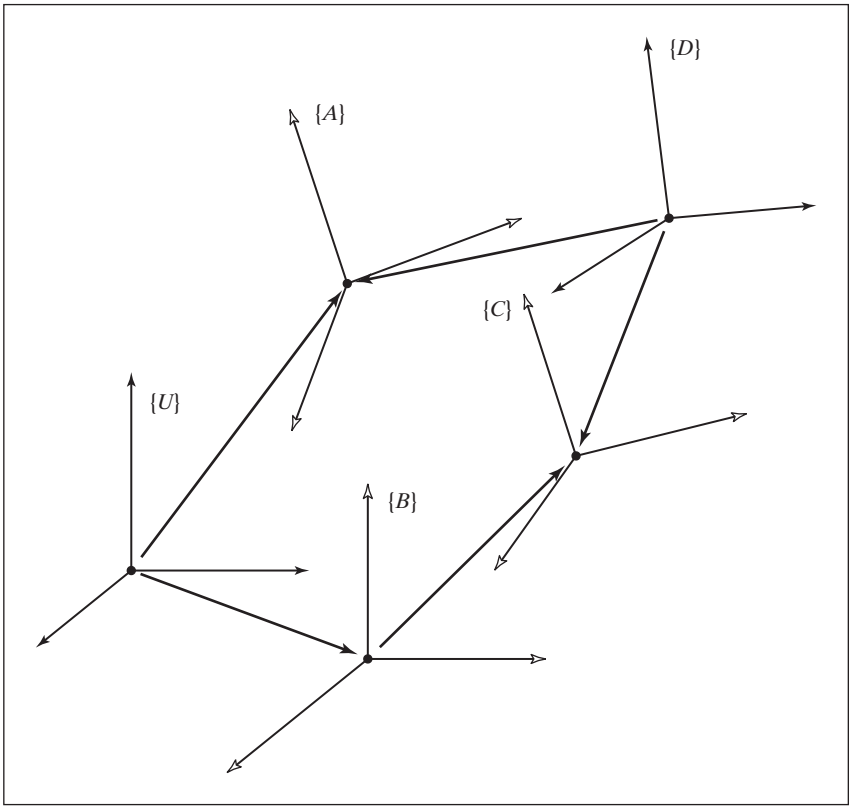

FIGURE 2.15: Example of a transform equation.

EXAMPLE 2.6

Assume that we know the transform ${}^{B}_{T}T$ in Fig. 2.16, which describes the frame at the manipulator's fingertips $\{T\}$ relative to the base of the manipulator, $\{B\}$, that we know where the tabletop is located in space relative to the manipulator's base (because we have a description of the frame $\{S\}$ that is attached to the table as shown, ${}^{B}_{S}T$), and that we know the location of the frame attached to the bolt lying on the table relative to the table frame—that is, ${}^{S}_{G}T$. Calculate the position and orientation of the bolt relative to the manipulator's hand, ${}^{T}_{G}T$.

Guided by our notation (and, it is hoped, our understanding), we compute the bolt frame relative to the hand frame as

$$ {}^{T}_{G}T = {}^{B}_{T}T^{-1}\ {}^{B}_{S}T\ {}^{S}_{G}T. \tag{2.55} $$

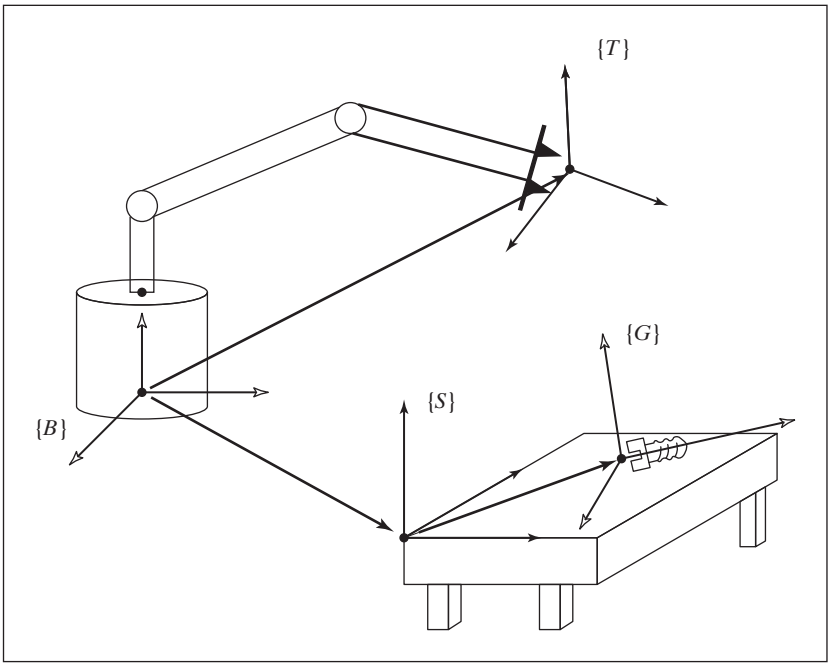

FIGURE 2.16: Manipulator reaching for a bolt.

2.8 MORE ON REPRESENTATION OF ORIENTATION

So far, our only means of representing an orientation is by giving a 3×3 rotation matrix. As shown, rotation matrices are special in that all columns are mutually orthogonal and have unit magnitude. Further, we will see that the determinant of a rotation matrix is always equal to $+1$. Rotation matrices may also be called **proper orthonormal matrices**, where "proper" refers to the fact that the determinant is $+1$ (nonproper orthonormal matrices have the determinant -1).

It is natural to ask whether it is possible to describe an orientation with fewer than nine numbers. A result from linear algebra (known as **Cayley's formula for orthonormal matrices** [3]) states that, for any proper orthonormal matrix R, there exists a skew-symmetric matrix S, such that

$$R = (I_3 - S)^{-1}(I_3 + S), \tag{2.56}$$

where I_3 is a 3×3 unit matrix. Now, a skew-symmetric matrix (i.e., $S = -S^T$) of dimension 3 is specified by three parameters (s_x, s_y, s_z) as

$$S = \begin{bmatrix} 0 & -s_z & s_y \\ s_z & 0 & -s_x \\ -s_y & s_x & 0 \end{bmatrix}. \tag{2.57}$$

Therefore, an immediate consequence of formula (2.56) is that any 3×3 rotation matrix can be specified by just three parameters.

Clearly, the nine elements of a rotation matrix are not all independent. In fact, given a rotation matrix, R, it is easy to write down the six dependencies between the elements. Imagine R as three columns, as originally introduced:

$$R = [\hat{X}\ \hat{Y}\ \hat{Z}]. \tag{2.58}$$

As we know from Section 2.2, these three vectors are the unit axes of some frame written in terms of the reference frame. Each is a unit vector, and all three must be mutually perpendicular, so we see that there are six constraints on the nine matrix elements:

$$\begin{aligned} |\hat{X}| &= 1, \\ |\hat{Y}| &= 1, \\ |\hat{Z}| &= 1, \\ \hat{X} \cdot \hat{Y} &= 0, \\ \hat{X} \cdot \hat{Z} &= 0, \\ \hat{Y} \cdot \hat{Z} &= 0. \end{aligned} \tag{2.59}$$

It is natural then to ask whether representations of orientation can be devised such that the representation is *conveniently* specified with three parameters. This section will present several such representations.

Whereas translations along three mutually perpendicular axes are quite easy to visualize, rotations seem less intuitive. Unfortunately, people have a hard time describing and specifying orientations in three-dimensional space. One difficulty is that rotations don't generally commute. That is, ${}^{A}_{B}R\ {}^{B}_{C}R$ is not the same as ${}^{B}_{C}R\ {}^{A}_{B}R$.

EXAMPLE 2.7

Consider two rotations, one about $\hat{Z}$ by 30 degrees, and one about $\hat{X}$ by 30 degrees:

$$R_z(30) = \begin{bmatrix} 0.866 & -0.500 & 0.000 \\ 0.500 & 0.866 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix} \tag{2.60}$$

$$R_x(30) = \begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.866 & -0.500 \\ 0.000 & 0.500 & 0.866 \end{bmatrix} \tag{2.61}$$

$$\begin{aligned} R_z(30)R_x(30) &= \begin{bmatrix} 0.87 & -0.43 & 0.25 \\ 0.50 & 0.75 & -0.43 \\ 0.00 & 0.50 & 0.87 \end{bmatrix} \\ \neq R_x(30)R_z(30) &= \begin{bmatrix} 0.87 & -0.50 & 0.00 \\ 0.43 & 0.75 & -0.50 \\ 0.25 & 0.43 & 0.87 \end{bmatrix} \end{aligned} \tag{2.62}$$

The fact that the order of rotations is important should not be surprising; furthermore, it is captured in the fact that we use matrices to represent rotations, because multiplication of matrices is not commutative in general.

Because rotations can be thought of either as operators or as descriptions of orientation, it is not surprising that different representations are favored for each of these uses. Rotation matrices are useful as operators. Their matrix form is such that, when multiplied by a vector, they perform the rotation operation. However, rotation matrices are somewhat unwieldy when used to specify an orientation. A human operator at a computer terminal who wishes to type in the specification of the desired orientation of a robot's hand would have a hard time inputting a nine-element matrix with orthonormal columns. A representation that requires only three numbers would be simpler. The following sections introduce several such representations.

X–Y–Z Fixed Angles

One method of describing the orientation of a frame $\{B\}$ is as follows:

> Start with the frame coincident with a known reference frame $\{A\}$. Rotate $\{B\}$ first about $\hat{X}_A$ by an angle γ, then about $\hat{Y}_A$ by an angle β, and, finally, about $\hat{Z}_A$ by an angle α.

Each of the three rotations takes place about an axis in the fixed reference frame $\{A\}$. We will call this convention for specifying an orientation **X–Y–Z fixed angles**. The word "fixed" refers to the fact that the rotations are specified about the fixed (i.e., nonmoving) reference frame (see Fig. 2.17). Sometimes, this convention is referred to as **roll, pitch, yaw angles**, but care must be used, as this name is often given to other related but different conventions.

The derivation of the equivalent rotation matrix, ${}^A_B R_{XYZ}(\gamma, \beta, \alpha)$, is straightforward, because all rotations occur about axes of the reference frame; that is,

$$
\begin{aligned}
{}^A_B R_{XYZ}(\gamma, \beta, \alpha) &= R_Z(\alpha) R_Y(\beta) R_X(\gamma) \\
&= \begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix}, \qquad (2.63)
\end{aligned}
$$

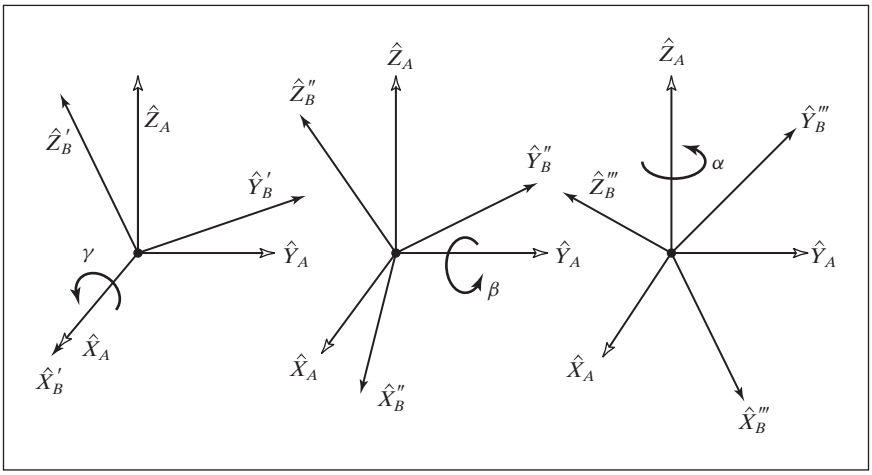

FIGURE 2.17: X–Y–Z fixed angles. Rotations are performed in the order $R_X(\gamma)$, $R_Y(\beta)$, $R_Z(\alpha)$.

where $c\alpha$ is shorthand for $\cos\alpha$, $s\alpha$ for $\sin\alpha$, and so on. It is extremely important to understand the order of rotations used in (2.63). Thinking in terms of rotations as operators, we have applied the rotations (from the *right*) of $R_X(\gamma)$, then $R_Y(\beta)$, and then $R_Z(\alpha)$. Multiplying (2.63) out, we obtain

$$
{}^{A}_{B}R_{XYZ}(\gamma,\beta,\alpha) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}. \tag{2.64}
$$

Keep in mind that the definition given here specifies the order of the three rotations. Equation (2.64) is correct only for rotations performed in the order: about $\hat{X}_A$ by γ, about $\hat{Y}_A$ by β, about $\hat{Z}_A$ by α.

The inverse problem, that of extracting equivalent X–Y–Z fixed angles from a rotation matrix, is often of interest. The solution depends on solving a set of transcendental equations: there are nine equations and three unknowns if (2.64) is equated to a given rotation matrix. Among the nine equations are six dependencies, so, essentially, we have three equations and three unknowns. Let

$$
{}^{A}_{B}R_{XYZ}(\gamma,\beta,\alpha) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{2.65}
$$

From (2.64), we see that, by taking the square root of the sum of the squares of r_{11} and r_{21}, we can compute $\cos\beta$. Then, we can solve for β with the arc tangent of $-r_{31}$ over the computed cosine. Then, as long as $c\beta \neq 0$, we can solve for α by taking the arc tangent of $r_{21}/c\beta$ over $r_{11}/c\beta$, and we can solve for γ by taking the arc tangent of $r_{32}/c\beta$ over $r_{33}/c\beta$.

In summary,

$$
\begin{aligned}
\beta &= \text{Atan2}(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}), \\
\alpha &= \text{Atan2}(r_{21}/c\beta, r_{11}/c\beta), \\
\gamma &= \text{Atan2}(r_{32}/c\beta, r_{33}/c\beta),
\end{aligned} \tag{2.66}
$$

where $\text{Atan2}(y, x)$ is a two-argument arc tangent function.[3]

Although a second solution exists, by using the positive square root in the formula for β, we always compute the single solution for which $-90.0° \leq \beta \leq 90.0°$. This is usually a good practice, because we can then define one-to-one mapping functions between various representations of orientation. However, in some cases, calculating all solutions is important (more on this will be presented in Chapter 4). If $\beta = \pm 90.0°$ (so that $c\beta = 0$), the solution of (2.67) degenerates. In those cases, only the sum or the difference of α and γ can be computed. One possible convention is to choose $\alpha = 0.0$ in these cases, which has the results given next.

[3] $\text{Atan2}(y, x)$ computes $\tan^{-1}(\frac{y}{x})$ but uses the signs of both x and y to identify the quadrant in which the resulting angle lies. For example, $\text{Atan2}(-2.0, -2.0) = -135°$, whereas $\text{Atan2}(2.0, 2.0) = 45°$, a distinction which would be lost with a single-argument arc tangent function. We are frequently computing angles that can range over a full 360°, so we will make use of the Atan2 function regularly. Note that Atan2 becomes undefined when both arguments are zero. It is sometimes called a "4-quadrant arc tangent," and some programming-language libraries have it predefined.

If $\beta = 90.0°$, then a solution can be calculated to be

$$\begin{aligned} \beta &= 90.0°, \\ \alpha &= 0.0, \\ \gamma &= \text{Atan2}(r_{12}, r_{22}). \end{aligned} \tag{2.67}$$

If $\beta = -90.0°$, then a solution can be calculated to be

$$\begin{aligned} \beta &= -90.0°, \\ \alpha &= 0.0, \\ \gamma &= -\text{Atan2}(r_{12}, r_{22}). \end{aligned} \tag{2.68}$$

Z–Y–X Euler Angles

Another possible description of a frame $\{B\}$ is as follows:

> Start with the frame coincident with a known frame $\{A\}$. Rotate $\{B\}$ first about $\hat{Z}_B$ by an angle α, then about $\hat{Y}_B$ by an angle β, and, finally, about $\hat{X}_B$ by an angle γ.

In this representation, each rotation is performed about an axis of the moving system $\{B\}$ rather than one of the fixed reference $\{A\}$. Such sets of three rotations are called **Euler angles**. Note that each rotation takes place about an axis whose location depends upon the preceding rotations. Because the three rotations occur about the axes $\hat{Z}$, $\hat{Y}$, and $\hat{X}$, we will call this representation **Z–Y–X Euler angles**.

Figure 2.18 shows the axes of $\{B\}$ after each Euler angle rotation is applied. Rotation α about $\hat{Z}$ causes $\hat{X}$ to rotate into $\hat{X}'$, $\hat{Y}$ to rotate into $\hat{Y}'$, and so on. An additional "prime" gets added to each axis with each rotation. A rotation matrix which is parameterized by Z–Y–X Euler angles will be indicated by the notation

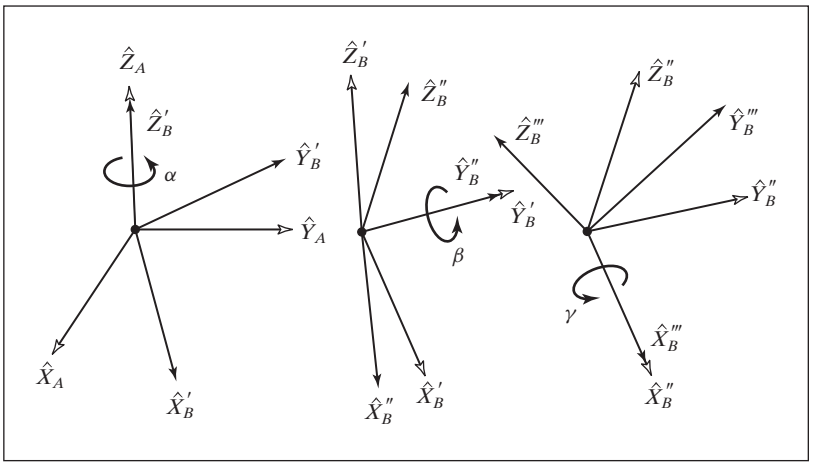

FIGURE 2.18: Z–Y–X Euler angles.

${}^A_B R_{Z'Y'X'}(\alpha, \beta, \gamma)$. Note that we have added "primes" to the subscripts to indicate that this rotation is described by Euler angles.

With reference to Fig. 2.18, we can use the intermediate frames $\{B'\}$ and $\{B''\}$ in order to give an expression for ${}^A_B R_{Z'Y'X'}(\alpha, \beta, \gamma)$. Thinking of the rotations as descriptions of these frames, we can immediately write

$$ {}^A_B R = {}^A_{B'} R \; {}^{B'}_{B''} R \; {}^{B''}_{B} R, \tag{2.69} $$

where each of the relative descriptions on the right-hand side of (2.69) is given by the statement of the Z–Y–X Euler angle convention. Namely, the final orientation of $\{B\}$ is given relative to $\{A\}$ as

$$ \begin{aligned} {}^A_B R_{Z'Y'X'} &= R_Z(\alpha) R_Y(\beta) R_X(\gamma) \\ &= \begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix}, \end{aligned} \tag{2.70} $$

where $c\alpha = \cos\alpha$, $s\alpha = \sin\alpha$, and so on. Multiplying out, we obtain

$$ {}^A_B R_{Z'Y'X'}(\alpha, \beta, \gamma) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}. \tag{2.71} $$

Note that the result is exactly the same as that obtained for the same three rotations taken in the opposite order about fixed axes! This somewhat nonintuitive result holds in general: three rotations taken about fixed axes yield the same final orientation as the same three rotations taken in opposite order about the axes of the moving frame.

Because (2.71) is equivalent to (2.64), there is no need to repeat the solution for extracting Z–Y–X Euler angles from a rotation matrix. That is, (2.66) can also be used to solve for Z–Y–X Euler angles that correspond to a given rotation matrix.

Z–Y–Z Euler Angles

Another possible description of a frame $\{B\}$ is

> Start with the frame coincident with a known frame $\{A\}$. Rotate $\{B\}$ first about $\hat{Z}_B$ by an angle α, then about $\hat{Y}_B$ by an angle β, and, finally, about $\hat{Z}_B$ by an angle γ.

Rotations are described relative to the frame we are moving, namely, $\{B\}$, so this is an Euler angle description. Because the three rotations occur about the axes $\hat{Z}$, $\hat{Y}$, and $\hat{Z}$, we will call this representation **Z–Y–Z Euler angles**.

Following the development exactly as in the last section, we arrive at the equivalent rotation matrix

$$ {}^A_B R_{Z'Y'Z'}(\alpha, \beta, \gamma) = \begin{bmatrix} c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta \\ s\alpha c\beta c\gamma + c\alpha s\gamma & -s\alpha c\beta s\gamma + c\alpha c\gamma & s\alpha s\beta \\ -s\beta c\gamma & s\beta s\gamma & c\beta \end{bmatrix}. \tag{2.72} $$

The solution for extracting Z–Y–Z Euler angles from a rotation matrix is stated next.

Given

$$ {}^{A}_{B}R_{Z'Y'Z'}(\alpha, \beta, \gamma) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{2.73} $$

then, if $\sin \beta \neq 0$, it follows that

$$ \begin{aligned} \beta &= \text{Atan2}(\sqrt{r_{31}^2 + r_{32}^2}, r_{33}), \\ \alpha &= \text{Atan2}(r_{23}/s\beta, r_{13}/s\beta), \\ \gamma &= \text{Atan2}(r_{32}/s\beta, -r_{31}/s\beta). \end{aligned} \tag{2.74} $$

Although a second solution exists (which we find by using the positive square root in the formula for β), we always compute the single solution for which $0.0 \leq \beta \leq 180.0°$. If $\beta = 0.0$ or $180.0°$, the solution of (2.74) degenerates. In those cases, only the sum or the difference of α and γ may be computed. One possible convention is to choose $\alpha = 0.0$ in these cases, which has the results given next.

If $\beta = 0.0$, then a solution can be calculated to be

$$ \begin{aligned} \beta &= 0.0, \\ \alpha &= 0.0, \\ \gamma &= \text{Atan2}(-r_{12}, r_{11}). \end{aligned} \tag{2.75} $$

If $\beta = 180.0°$, then a solution can be calculated to be

$$ \begin{aligned} \beta &= 180.0°, \\ \alpha &= 0.0, \\ \gamma &= \text{Atan2}(r_{12}, -r_{11}). \end{aligned} \tag{2.76} $$

Other Angle-Set Conventions

In the preceding subsections, we have seen three conventions for specifying orientation: X–Y–Z fixed angles, Z–Y–X Euler angles, and Z–Y–Z Euler angles. Each of these conventions requires performing three rotations about principal axes in a certain order. These conventions are examples of a set of 24 conventions that we will call **angle-set conventions**. Of these, 12 conventions are for fixed-angle sets, and 12 are for Euler angle sets. Note that, because of the duality of fixed-angle sets with Euler angle sets, there are really only 12 unique parameterizations of a rotation matrix by using successive rotations about principal axes. There is often no particular reason to favor one convention over another, but various authors adopt different ones, so it is useful to list the equivalent rotation matrices for all 24 conventions. Appendix B (in the back of the book) gives the equivalent rotation matrices for all 24 conventions.

Equivalent Angle–Axis Representation

With the notation $R_X(30.0)$, we give the description of an orientation by giving an axis, $\hat{X}$, and an angle, 30.0 degrees. This is an example of an **equivalent angle–axis** representation. If the axis is a *general* direction (rather than one of the unit directions)

any orientation may be obtained through proper axis and angle selection. Consider the following description of a frame $\{B\}$:

> Start with the frame coincident with a known frame $\{A\}$; then rotate $\{B\}$ about the vector ${}^A\hat{K}$ by an angle θ according to the right-hand rule.

Vector $\hat{K}$ is sometimes called the equivalent axis of a finite rotation. A general orientation of $\{B\}$ relative to $\{A\}$ may be written as ${}^A_B R(\hat{K}, \theta)$ or $R_K(\theta)$, and will be called the equivalent angle–axis representation.[4] The specification of the vector ${}^A\hat{K}$ requires only two parameters, because its length is always taken to be one. The angle specifies a third parameter. Often, we will multiply the unit direction, $\hat{K}$, with the amount of rotation, θ, to form a compact 3×1 vector description of orientation, denoted by K (no "hat") (see Fig. 2.19).

When the axis of rotation is chosen from among the principal axes of $\{A\}$, then the equivalent rotation matrix takes on the familiar form of planar rotations:

$$R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \tag{2.77}$$

$$R_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{2.78}$$

$$R_Z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.79}$$

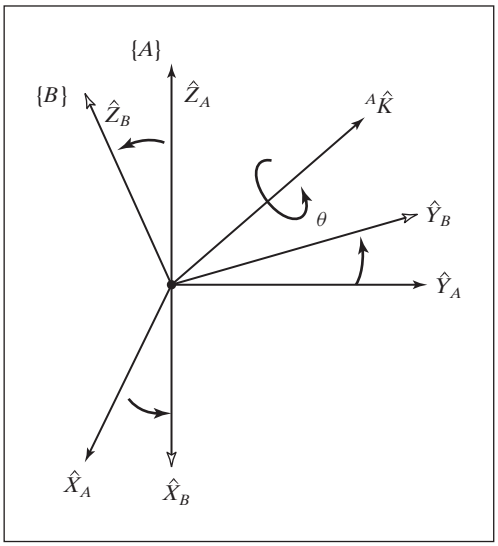

FIGURE 2.19: Equivalent angle–axis representation.

[4] That such a $\hat{K}$ and θ exist for any orientation of $\{B\}$ relative to $\{A\}$ was shown originally by Euler, and is known as Euler's theorem on rotation [3].

If the axis of rotation is a general axis, it can be shown (as in Exercise 2.6) that the equivalent rotation matrix is

$$R_K(\theta) = \begin{bmatrix} k_x k_x v\theta + c\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y k_y v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z k_z v\theta + c\theta \end{bmatrix}, \tag{2.80}$$

where $c\theta = \cos\theta$, $s\theta = \sin\theta$, $v\theta = 1 - \cos\theta$, and ${}^A\hat{K} = [k_x k_y k_z]^T$. The sign of θ is determined by the right-hand rule, with the thumb pointing along the positive sense of ${}^A\hat{K}$.

Equation (2.80) converts from angle–axis representation to rotation-matrix representation. Note that, given any axis of rotation and any angular amount, we can easily construct an equivalent rotation matrix.

The inverse problem, namely, that of computing $\hat{K}$ and θ from a given rotation matrix, is mostly left for the exercises (Exercises 2.6 and 2.7), but a partial result is given here [3]. If

$${}^A_B R_K(\theta) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{2.81}$$

then

$$\theta = \text{Acos}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

and

$$\hat{K} = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}. \tag{2.82}$$

This solution always computes a value of θ between 0 and 180 degrees. For any axis–angle pair $({}^A\hat{K}, \theta)$, there is another pair, namely, $(-{}^A\hat{K}, -\theta)$, which results in the same orientation in space, with the same rotation matrix describing it. Therefore, in converting from a rotation-matrix into an angle–axis representation, we are faced with choosing between solutions. A more serious problem is that, for small angular rotations, the axis becomes ill-defined. Clearly, if the amount of rotation goes to zero, the axis of rotation becomes completely undefined. The solution given by (2.82) fails if $\theta = 0°$ or $\theta = 180°$.

EXAMPLE 2.8

A frame $\{B\}$ is described as initially coincident with $\{A\}$. We then rotate $\{B\}$ about the vector ${}^A\hat{K} = [0.7070 \quad 7070 \quad 0]^T$ (passing through the origin) by an amount $\theta = 30$ degrees. Give the frame description of $\{B\}$.

Substituting into (2.80) yields the rotation-matrix part of the frame description. There was no translation of the origin, so the position vector is $[0, 0, 0]^T$. Hence,

$${}^A_B T = \begin{bmatrix} 0.933 & 0.067 & 0.354 & 0.0 \\ 0.067 & 0.933 & -0.354 & 0.0 \\ -0.354 & 0.354 & 0.866 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{2.83}$$
