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 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 they are defined within. 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, are 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 be *oriented* arbitrarily while keeping the point between the fingertips at the same

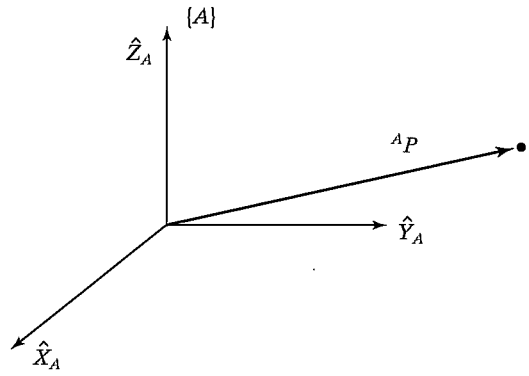

FIGURE 2.1: Vector relative to frame (example).

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

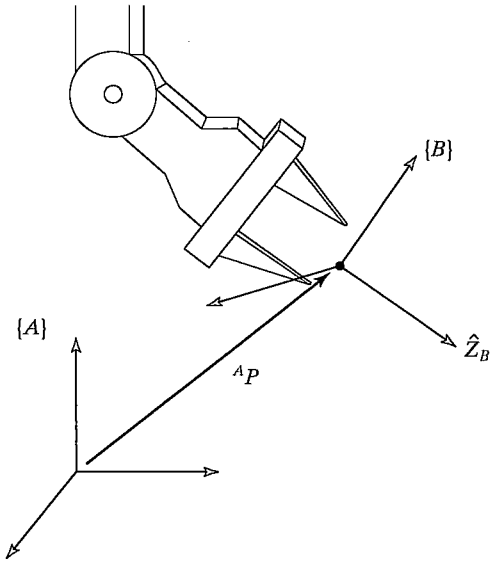

FIGURE 2.2: Locating an object in position and orientation.

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 ${}^A_BR$ (the choice of leading sub- and superscripts in the definition of rotation matrices will become clear in following sections):

$$ {}^A_BR = \left[{}^A\hat{X}_B \;\; {}^A\hat{Y}_B \;\; {}^A\hat{Z}_B \right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{2.2} $$

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

[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.)

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:

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

For brevity, 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,

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

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

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

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

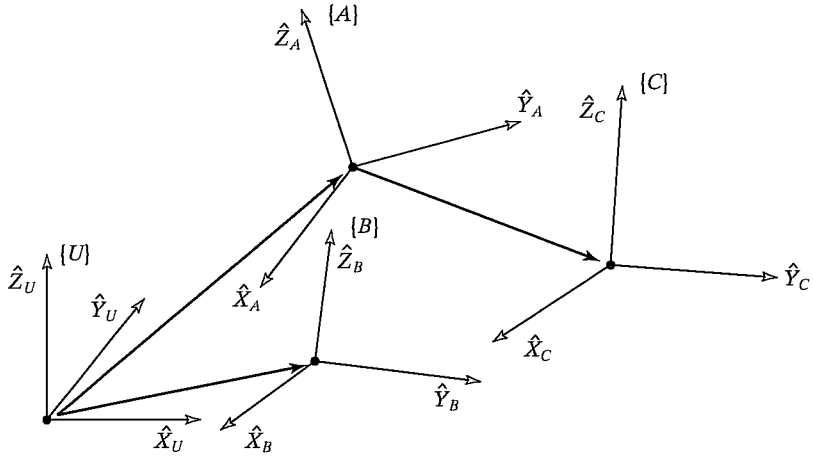

FIGURE 2.3: Example of several frames.

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, 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 $\{A\}$.

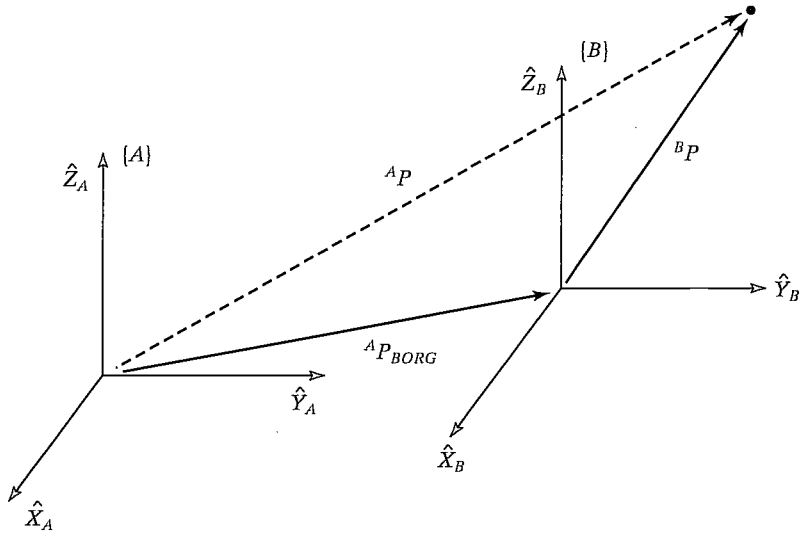

FIGURE 2.4: Translational mapping.

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:

$$ {}^{A}_{B}R = \left[{}^{A}\hat{X}_{B} \;\; {}^{A}\hat{Y}_{B} \;\; {}^{A}\hat{Z}_{B} \right] = \begin{bmatrix} {}^{B}\hat{X}^{T}_{A} \\ {}^{B}\hat{Y}^{T}_{A} \\ {}^{B}\hat{Z}^{T}_{A} \end{bmatrix}. \tag{2.11} $$

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.

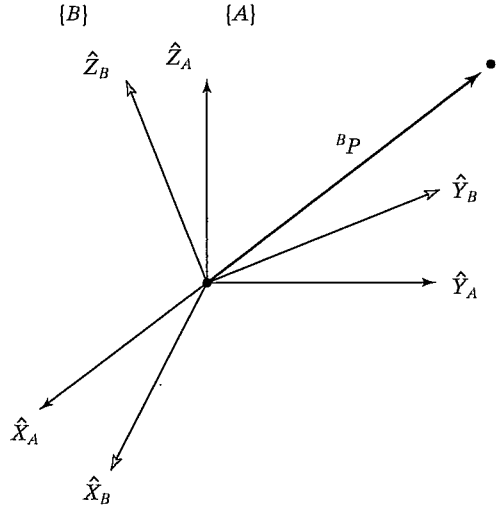

FIGURE 2.5: Rotating the description of a vector.

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

$$\begin{aligned} {}^A px &= {}^B\hat{X}_A \cdot {}^B P, \\ {}^A p_y &= {}^B\hat{Y}_A \cdot {}^B P, \\ {}^A p_z &= {}^B\hat{Z}_A \cdot {}^B P. \end{aligned} \tag{2.12}$$

In order to express (2.13) 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 (2.13) may be written compactly, by using a rotation matrix, as

$${}^A P = {}^A_B R \, {}^B P. \tag{2.13}$$

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

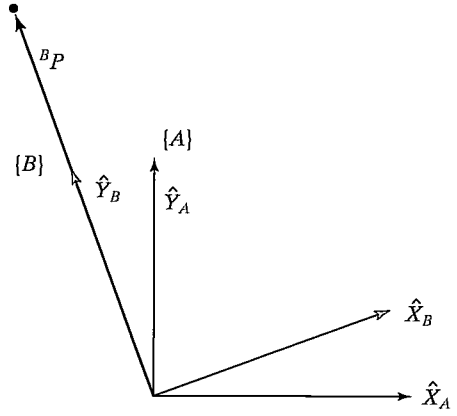

FIGURE 2.6: $\{B\}$ rotated 30 degrees about $\hat{Z}$.

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

$$ {}^{A}_{B}R = \begin{bmatrix} 0.866 & -0.500 & 0.000 \\ 0.500 & 0.866 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}. \tag{2.14} $$

Given

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

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

$$ {}^{A}P = {}^{A}_{B}R\,{}^{B}P = \begin{bmatrix} -1.000 \\ 1.732 \\ 0.000 \end{bmatrix}. \tag{2.16} $$

Here, ${}^{A}_{B}R$ acts as a mapping that is used to describe ${}^{B}P$ relative to frame $\{A\}$, ${}^{A}P$. 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 ${}^{A}P_{BORG}$. Also $\{B\}$ is rotated with respect to $\{A\}$, as described by ${}^{A}_{B}R$. Given ${}^{B}P$, we wish to compute ${}^{A}P$, as in Fig. 2.7.

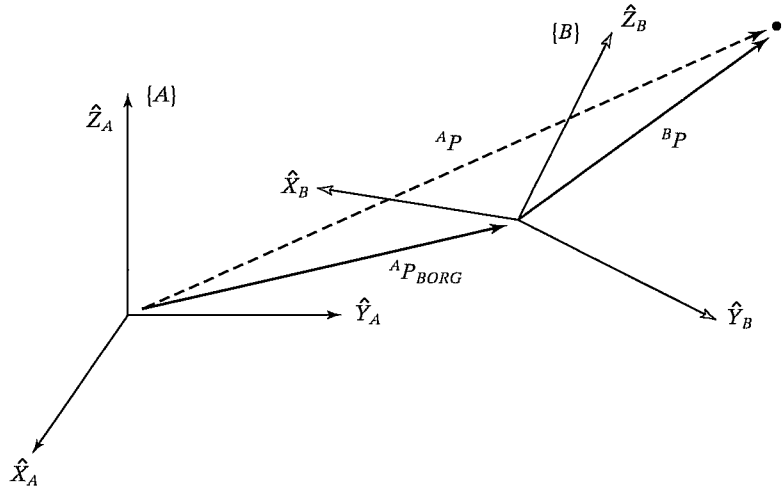

FIGURE 2.7: General transform of a vector.

We can first change ${}^{B}P$ 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 ${}^{A}_{B}R$ as in the last section. We then account for the translation between origins by simple vector addition, as before, and obtain

$$ {}^{A}P = {}^{A}_{B}R\,{}^{B}P + {}^{A}P_{BORG}. \tag{2.17} $$

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

$$ {}^{A}P = {}^{A}_{B}T\,{}^{B}P. \tag{2.18} $$

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 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

$$ \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} = \left[\begin{array}{c|c} {}^{A}_{B}R & {}^{A}P_{BORG} \\ \hline 0\ \ 0\ \ 0 & 1 \end{array}\right] \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix}. \tag{2.19} $$

In other words,

1. a "1" is added as the last element of the 4×1 vectors;
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

$$ \begin{aligned} {}^{A}P &= {}^{A}_{B}R\,{}^{B}P + {}^{A}P_{BORG} \\ 1 &= 1. \end{aligned} \tag{2.20} $$

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 $\hat{Z}$ by 30 degrees, translated 10 units in $\hat{X}_A$, and translated 5 units in $\hat{Y}_A$. Find ${}^{A}P$, where ${}^{B}P = [3.07.00.0]^T$.

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

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

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

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

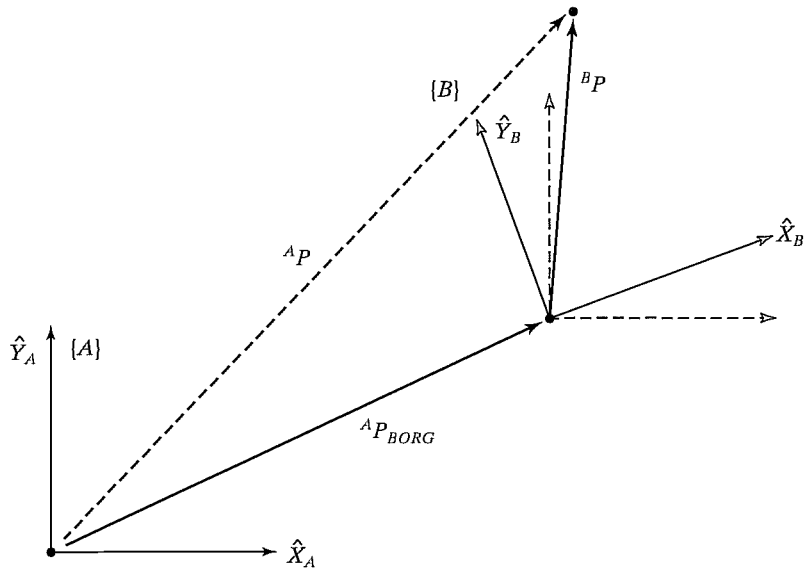

FIGURE 2.8: Frame $\{B\}$ rotated and translated.

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 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

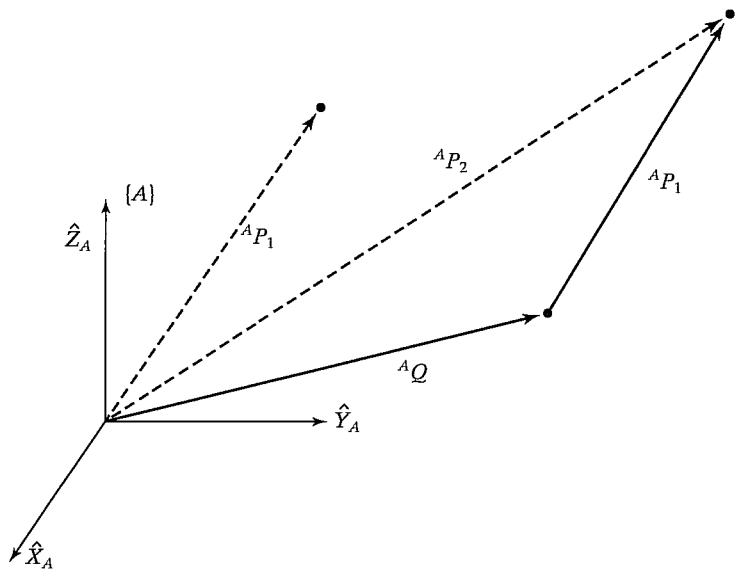

FIGURE 2.9: Translation operator.

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) is 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 $\hat{K}$.

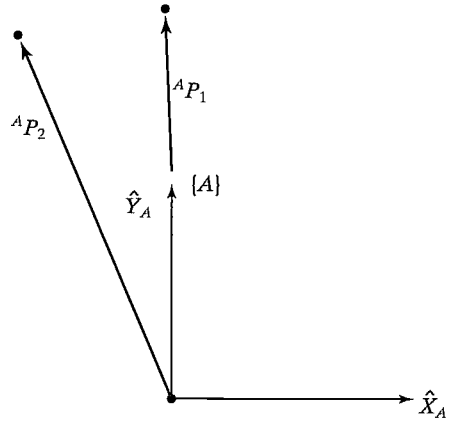

FIGURE 2.10: The vector ${}^{A}P_1$ rotated 30 degrees about $\hat{Z}$.

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. 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

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

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, and 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) is 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 $\hat{Z}$ 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

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

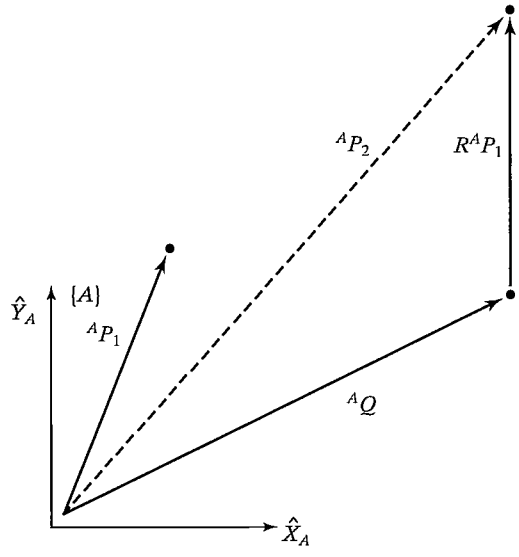

FIGURE 2.11: The vector ${}^{A}P_1$ rotated and translated to form ${}^{A}P_2$.

Given

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

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 that 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 ${}^{A}P$.

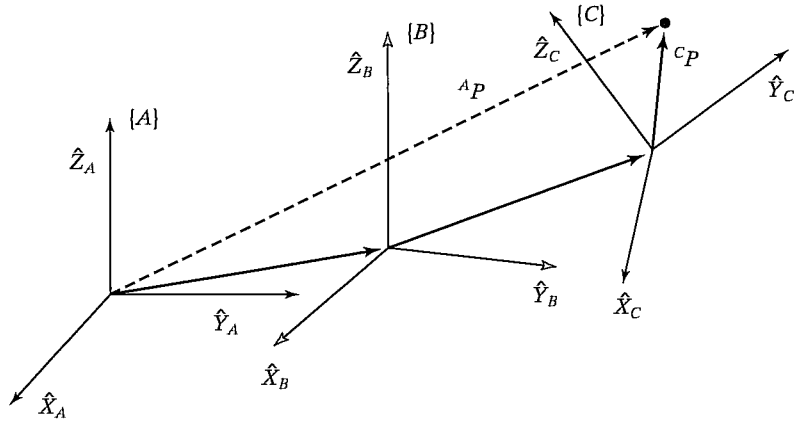

FIGURE 2.12: Compound frames: Each is known relative to the previous one.

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

$$ {}^A_C T = {}^A_B T {}^B_C T. \tag{2.40} $$

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

$$ {}^A_C T = \left[\begin{array}{ccc|c} & {}^A_B R \, {}^B_C R & & {}^A_B R \, {}^B P_{CORG} + {}^A P_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.41} $$

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 $\hat{Z}$ 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

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

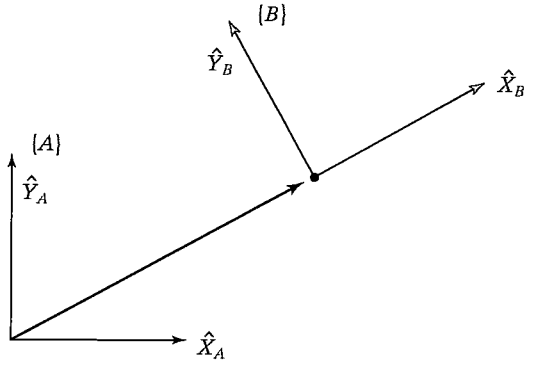

FIGURE 2.13: $\{B\}$ relative to $\{A\}$.

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**:

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

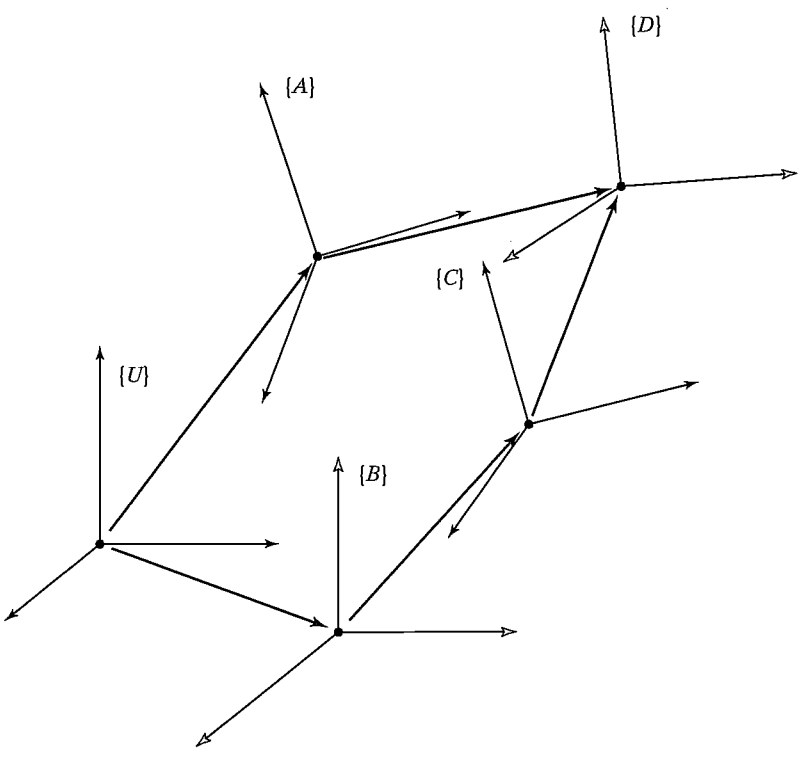

FIGURE 2.14: Set of transforms forming a loop.

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 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

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

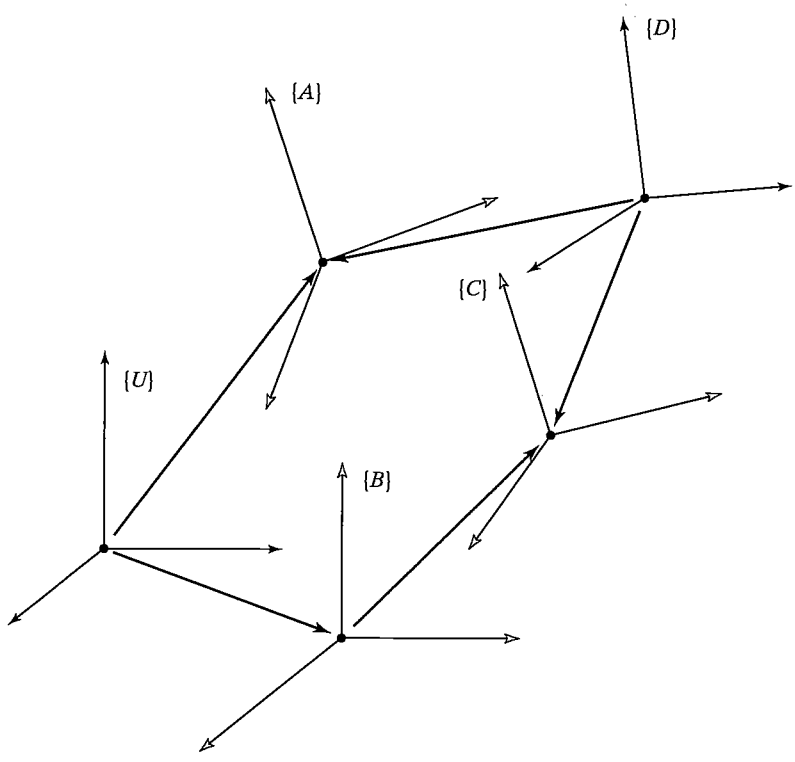

FIGURE 2.15: Example of a transform equation.

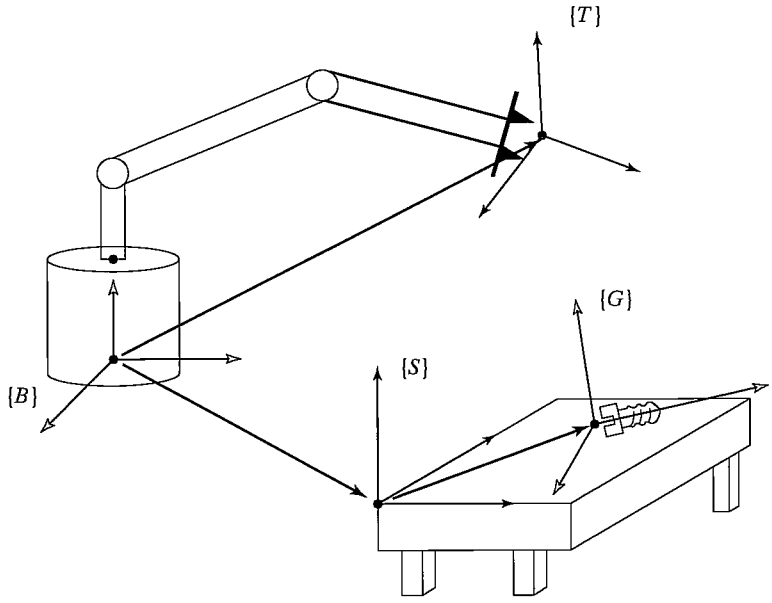

FIGURE 2.16: Manipulator reaching for a bolt.

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

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

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

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

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

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

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:

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

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

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

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

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 (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.

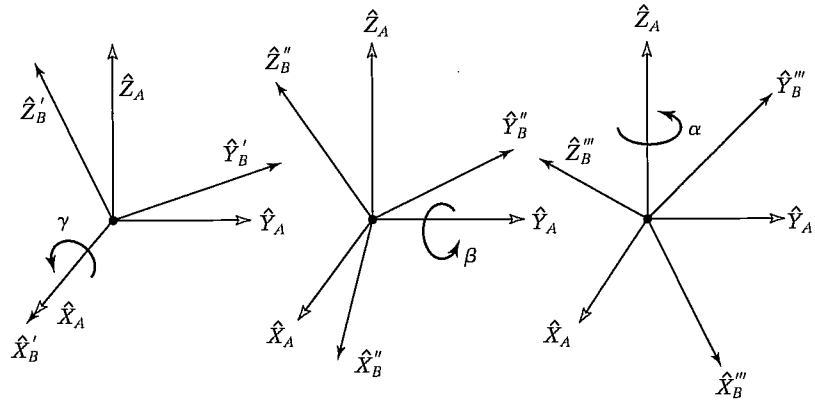

FIGURE 2.17: X–Y–Z fixed angles. Rotations are performed in the order $R_X(\gamma)$, $R_Y(\beta)$, $R_Z(\alpha)$.

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,

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

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 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.

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

[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, $\text{Atan 2}(-2.0, -2.0) = -135°$, whereas $\text{Atan 2}(2.0, 2.0) = 45°$, 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.

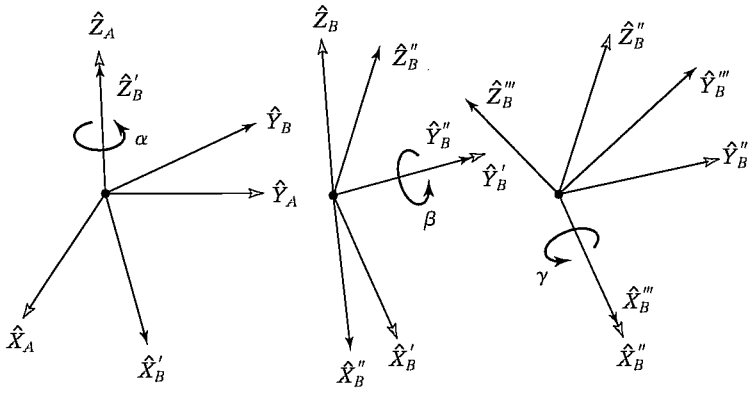

FIGURE 2.18: Z–Y–X Euler angles.

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 ${}^A_B R_{Z'Y'X'}(\alpha, \beta, \gamma)$. 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 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:

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

[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].

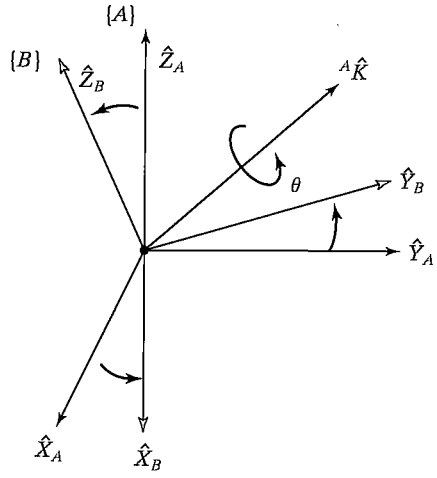

FIGURE 2.19: Equivalent angle–axis representation.

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

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

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 ${}^A\hat{K} = [0.7070\ 7070\ 0]^T$ (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,

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

Up to this point, all rotations we have discussed have been about axes that pass through the origin of the reference system. If we encounter a problem for which this is not true, we can reduce the problem to the "axis through the origin" case by defining additional frames whose origins lie on the axis and then solving a transform equation.

EXAMPLE 2.9

A frame $\{B\}$ is described as initially coincident with $\{A\}$. We then rotate $\{B\}$ about the vector ${}^A\hat{K} = [0.707\ 0.707\ 0.0]^T$ (passing through the point ${}^AP = [1.0\ 2.0\ 3.0]$) by an amount $\theta = 30$ degrees. Give the frame description of $\{B\}$.

Before the rotation, $\{A\}$ and $\{B\}$ are coincident. As is shown in Fig. 2.20, we define two new frames, $\{A'\}$ and $\{B'\}$, which are coincident with each other and have the same orientation as $\{A\}$ and $\{B\}$ respectively, but are translated relative to $\{A\}$ by an offset that places their origins on the axis of rotation. We will choose

$${}^A_{A'} T = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 2.0 \\ 0.0 & 0.0 & 1.0 & 3.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{2.84}$$

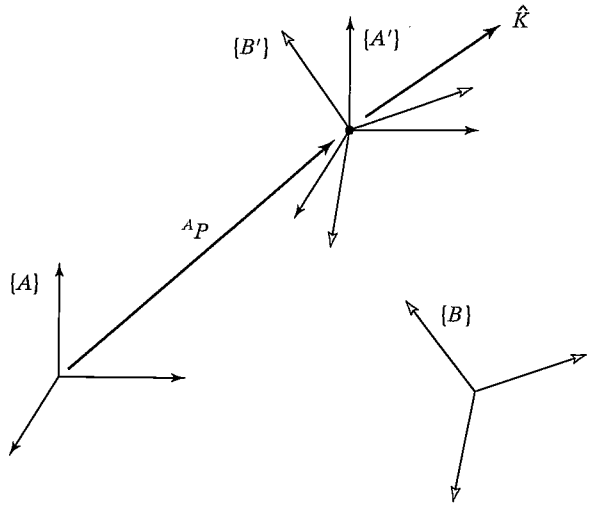

FIGURE 2.20: Rotation about an axis that does not pass through the origin of $\{A\}$. Initially, $\{B\}$ was coincident with $\{A\}$.

Similarly, the description of $\{B\}$ in terms of $\{B'\}$ is

$$ {}^{B'}_{B}T = \begin{bmatrix} 1.0 & 0.0 & 0.0 & -1.0 \\ 0.0 & 1.0 & 0.0 & -2.0 \\ 0.0 & 0.0 & 1.0 & -3.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{2.85} $$

Now, keeping other relationships fixed, we can rotate $\{B'\}$ relative to $\{A'\}$. This is a rotation about an axis that passes through the origin, so we can use (2.80) to compute $\{B'\}$ relative to $\{A'\}$. 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 $[0, 0, 0]^T$. Thus, we have

$$ {}^{A'}_{B'}T = \begin{bmatrix} 0.933 & 0.067 & 0.354 & 0.0 \\ 0.067 & 0.933 & -0.354 & 0.0 \\ -0.354 & 0.354 & 0.866 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{2.86} $$

Finally, we can write a transform equation to compute the desired frame,

$$ {}^{A}_{B}T = {}^{A}_{A'}T\, {}^{A'}_{B'}T\, {}^{B'}_{B}T, \tag{2.87} $$

which evaluates to

$$ {}^{A}_{B}T = \begin{bmatrix} 0.933 & 0.067 & 0.354 & -1.13 \\ 0.067 & 0.933 & -0.354 & 1.13 \\ -0.354 & 0.354 & 0.866 & 0.05 \\ 0.000 & 0.000 & 0.000 & 1.00 \end{bmatrix}. \tag{2.88} $$

A rotation about an axis that does not pass through the origin causes a change in position, plus the same final orientation as if the axis had passed through the origin.

Note that we could have used any definition of $\{A'\}$ and $\{B'\}$ such that their origins were on the axis of rotation. Our particular choice of orientation was arbitrary, and our choice of the position of the origin was one of an infinity of possible choices lying along the axis of rotation. (See also Exercise 2.14.)

Euler parameters

Another representation of orientation is by means of four numbers called the **Euler parameters**. Although complete discussion is beyond the scope of the book, we state the convention here for reference.

In terms of the equivalent axis $\hat{K} = [k_x \; k_y \; k_z]^T$ and the equivalent angle θ, the Euler parameters are given by

$$\begin{aligned} \epsilon_1 &= k_x \sin\frac{\theta}{2}, \\ \epsilon_2 &= k_y \sin\frac{\theta}{2}, \\ \epsilon_3 &= k_z \sin\frac{\theta}{2}, \\ \epsilon_4 &= \cos\frac{\theta}{2}. \end{aligned} \tag{2.89}$$

It is then clear that these four quantities are not independent:

$$\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \epsilon_4^2 = 1 \tag{2.90}$$

must always hold. Hence, an orientation might be visualized as a point on a unit hypersphere in four-dimensional space.

Sometimes, the Euler parameters are viewed as a 3×1 vector plus a scalar. However, as a 4×1 vector, the Euler parameters are known as a **unit quaternion**.

The rotation matrix R_ϵ that is equivalent to a set of Euler parameters is

$$R_\epsilon = \begin{bmatrix} 1-2\epsilon_2^2-2\epsilon_3^2 & 2(\epsilon_1\epsilon_2-\epsilon_3\epsilon_4) & 2(\epsilon_1\epsilon_3+\epsilon_2\epsilon_4) \\ 2(\epsilon_1\epsilon_2+\epsilon_3\epsilon_4) & 1-2\epsilon_1^2-2\epsilon_3^2 & 2(\epsilon_2\epsilon_3-\epsilon_1\epsilon_4) \\ 2(\epsilon_1\epsilon_3-\epsilon_2\epsilon_4) & 2(\epsilon_2\epsilon_3+\epsilon_1\epsilon_4) & 1-2\epsilon_1^2-2\epsilon_2^2 \end{bmatrix}. \tag{2.91}$$

Given a rotation matrix, the equivalent Euler parameters are

$$\begin{aligned} \epsilon_1 &= \frac{r_{32}-r_{23}}{4\epsilon_4}, \\ \epsilon_2 &= \frac{r_{13}-r_{31}}{4\epsilon_4}, \\ \epsilon_3 &= \frac{r_{21}-r_{12}}{4\epsilon_4}, \\ \epsilon_4 &= \frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}}. \end{aligned} \tag{2.92}$$

Note that (2.92) is not useful in a computational sense if the rotation matrix represents a rotation of 180 degrees about some axis, because ϵ_4 goes to zero. However, it can be shown that, in the limit, all the expressions in (2.92) remain finite even for this case. In fact, from the definitions in (2.88), it is clear that all ϵ_i remain in the interval $[-1, 1]$.

Taught and predefined orientations

In many robot systems, it will be possible to "teach" positions and orientations by using the robot itself. The manipulator is moved to a desired location, and this position is recorded. A frame taught in this manner need not necessarily be one to which the robot will be commanded to return; it could be a part location or a fixture location. In other words, the robot is used as a measuring tool having six degrees of freedom. Teaching an orientation like this completely obviates the need for the human programmer to deal with orientation representation at all. In the computer, the taught point is stored as a rotation matrix (or however), but the user never has to see or understand it. Robot systems that allow teaching of frames by using the robot are thus highly recommended.

Besides teaching frames, some systems have a set of predefined orientations, such as "pointing down" or "pointing left." These specifications are very easy for humans to deal with. However, if this were the only means of describing and specifying orientation, the system would be very limited.

2.9 TRANSFORMATION OF FREE VECTORS

We have been concerned mostly with position vectors in this chapter. In later chapters, we will discuss velocity and force vectors as well. These vectors will transform differently because they are a different *type* of vector.

In mechanics, one makes a distinction between the equality and the equivalence of vectors. *Two vectors are equal if they have the same dimensions, magnitude, and direction.* Two vectors that are considered *equal* could have different lines of action—for example, the three equal vectors in Fig 2.21. These velocity vectors have the same dimensions, magnitude, and direction and so are equal according to our definition.

Two vectors are equivalent in a certain capacity if each produces the very same effect in this capacity. Thus, if the criterion in Fig. 2.21 is distance traveled, all three vectors give the same result and are thus equivalent in this capacity. If the criterion is height above the xy plane, then the vectors are not equivalent despite their equality. Thus, relationships between vectors and notions of equivalence *depend entirely on the situation at hand.* Furthermore, vectors that are not equal might cause equivalent effects in certain cases.

We will define two basic classes of vector quantities that might be helpful.

The term **line vector** refers to a vector that is dependent on its **line of action**, along with direction and magnitude, for causing its effects. Often, the effects of a force vector depend upon its line of action (or point of application), so it would then be considered a line vector.

A **free vector** refers to a vector that may be positioned anywhere in space without loss or change of meaning, provided that magnitude and direction are preserved.

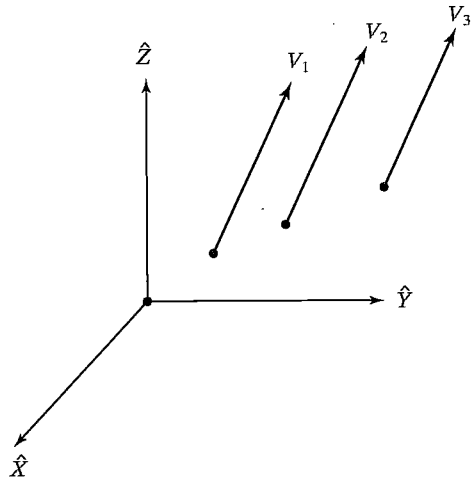

FIGURE 2.21: Equal velocity vectors.

For example, a pure moment vector is always a free vector. If we have a moment vector ${}^{B}N$ that is known in terms of $\{B\}$, then we calculate the same moment in terms of frame $\{A\}$ as

$$ {}^{A}N = {}^{A}_{B}R\,{}^{B}N. \tag{2.93} $$

In other words, all that counts is the magnitude and direction (in the case of a free vector), so only the rotation matrix relating the two systems is used in transforming. The relative locations of the origins do not enter into the calculation.

Likewise, a velocity vector written in $\{B\}$, ${}^{B}V$, is written in $\{A\}$ as

$$ {}^{A}V = {}^{A}_{B}R\,{}^{B}V. \tag{2.94} $$

The velocity of a point is a free vector, so all that is important is its direction and magnitude. The operation of rotation (as in (2.94)) does not affect the magnitude, yet accomplishes the rotation that changes the description of the vector from $\{B\}$ to $\{A\}$. Note that ${}^{A}P_{BORG}$, which would appear in a position-vector transformation, does not appear in a velocity transform. For example, in Fig. 2.22, if ${}^{B}V = 5\hat{X}$, then ${}^{A}V = 5\hat{Y}$.

Velocity vectors and force and moment vectors will be introduced more fully in Chapter 5.

2.10 COMPUTATIONAL CONSIDERATIONS

The availability of inexpensive computing power is largely responsible for the growth of the robotics industry; yet, for some time to come, efficient computation will remain an important issue in the design of a manipulation system.

The homogeneous representation is useful as a conceptual entity, but transformation software typically used in industrial manipulation systems does not make use of it directly, because the time spent multiplying by zeros and ones is wasteful.

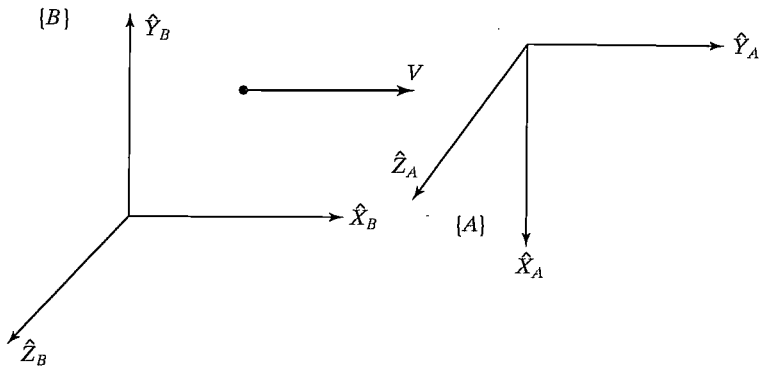

FIGURE 2.22: Transforming velocities.

Usually, the computations shown in (2.41) and (2.45) are performed, rather than the direct multiplication or inversion of 4×4 matrices.

The *order* in which transformations are applied can make a large difference in the amount of computation required to compute the same quantity. Consider performing multiple rotations of a vector, as in

$$^{A}P = {}^{A}_{B}R\,{}^{B}_{C}R\,{}^{C}_{D}R\,{}^{D}P. \tag{2.95}$$

One choice is to first multiply the three rotation matrices together, to form ${}^{A}_{D}R$ in the expression

$$^{A}P = {}^{A}_{D}R\,{}^{D}P. \tag{2.96}$$

Forming ${}^{A}_{D}R$ from its three constituents requires 54 multiplications and 36 additions. Performing the final matrix-vector multiplication of (2.96) requires an additional 9 multiplications and 6 additions, bringing the totals to 63 multiplications and 42 additions.

If, instead, we transform the vector through the matrices one at a time, that is,

$$\begin{aligned} ^{A}P &= {}^{A}_{B}R\,{}^{B}_{C}R\,{}^{C}_{D}R\,{}^{D}P \\ ^{A}P &= {}^{A}_{B}R\,{}^{B}_{C}R\,{}^{C}P \\ ^{A}P &= {}^{A}_{B}R\,{}^{B}P \\ ^{A}P &= {}^{A}P, \end{aligned} \tag{2.97}$$

then the total computation requires only 27 multiplications and 18 additions, fewer than half the computations required by the other method.

Of course, in some cases, the relationships ${}^{A}_{B}R$, ${}^{B}_{C}R$, and ${}^{C}_{D}R$ are constant, while there are many ${}^{D}P_i$ that need to be transformed into ${}^{A}P_i$. In such a case, it is more efficient to calculate ${}^{A}_{D}R$ once, and then use it for all future mappings. See also Exercise 2.16.

EXAMPLE 2.10

Give a method of computing the product of two rotation matrices, ${}^A_B R\, {}^B_C R$, that uses fewer than 27 multiplications and 18 additions.

Where $\hat{L}_i$ are the columns of ${}^B_C R$ and $\hat{C}_i$ are the three columns of the result, compute

$$\begin{aligned} \hat{C}_1 &= {}^A_B R \hat{L}_i, \\ \hat{C}_2 &= {}^A_B R \hat{L}_2, \\ \hat{C}_3 &= \hat{C}_1 \times \hat{C}_2, \end{aligned} \tag{2.98}$$

which requires 24 multiplications and 15 additions.

BIBLIOGRAPHY

[1] B. Noble, *Applied Linear Algebra*, Prentice-Hall, Englewood Cliffs, NJ, 1969.

[2] D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[3] O. Bottema and B. Roth, *Theoretical Kinematics*, North Holland, Amsterdam, 1979.

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

[5] I. Shames, *Engineering Mechanics*, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[6] Symon, *Mechanics*, 3rd edition, Addison-Wesley, Reading, MA, 1971.

[7] B. Gorla and M. Renaud, *Robots Manipulateurs*, Cepadues-Editions, Toulouse, 1984.

EXERCISES

2.1 [15] A vector ${}^A P$ is rotated about $\hat{Z}_A$ by θ degrees and is subsequently rotated about $\hat{X}_A$ by ϕ degrees. Give the rotation matrix that accomplishes these rotations in the given order.

2.2 [15] A vector ${}^A P$ is rotated about $\hat{Y}_A$ by 30 degrees and is subsequently rotated about $\hat{X}_A$ by 45 degrees. Give the rotation matrix that accomplishes these rotations in the given order.

2.3 [16] A frame $\{B\}$ is located initially coincident with a frame $\{A\}$. We rotate $\{B\}$ about $\hat{Z}_B$ by θ degrees, and then we rotate the resulting frame about $\hat{X}_B$ by ϕ degrees. Give the rotation matrix that will change the descriptions of vectors from ${}^B P$ to ${}^A P$.

2.4 [16] A frame $\{B\}$ is located initially coincident with a frame $\{A\}$. We rotate $\{B\}$ about $\hat{Z}_B$ by 30 degrees, and then we rotate the resulting frame about $\hat{X}_B$ by 45 degrees. Give the rotation matrix that will change the description of vectors from ${}^B P$ to ${}^A P$.

2.5 [13] ${}^A_B R$ is a 3×3 matrix with eigenvalues $1, e^{+ai}$, and e^{-ai}, where $i = \sqrt{-1}$. What is the physical meaning of the eigenvector of ${}^A_B R$ associated with the eigenvalue 1?

2.6 [21] Derive equation (2.80).

2.7 [24] Describe (or program) an algorithm that extracts the equivalent angle and axis of a rotation matrix. Equation (2.82) is a good start, but make sure that your algorithm handles the special cases $\theta = 0°$ and $\theta = 180°$.

2.8 [29] Write a subroutine that changes representation of orientation from rotation-matrix form to equivalent angle–axis form. A Pascal-style procedure declaration would begin

```
Procedure RMTOAA (VAR R:mat33; VAR K:vec3; VAR theta: real);
```

Write another subroutine that changes from equivalent angle–axis representation to rotation-matrix representation:

```
Procedure AATORM(VAR K:vec3; VAR theta: real: VAR R:mat33);
```

Write the routines in C if you prefer.
Run these procedures on several cases of test data back-to-back and verify that you get back what you put in. Include some of the difficult cases!

2.9 [27] Do Exercise 2.8 for roll, pitch, yaw angles about fixed axes.

2.10 [27] Do Exercise 2.8 for Z–Y–Z Euler angles.

2.11 [10] Under what condition do two rotation matrices representing finite rotations commute? A proof is not required.

2.12 [14] A velocity vector is given by

$$ {}^{B}V = \begin{bmatrix} 10.0 \\ 20.0 \\ 30.0 \end{bmatrix}. $$

Given

$$ {}^{A}_{B}T = \begin{bmatrix} 0.866 & -0.500 & 0.000 & 11.0 \\ 0.500 & 0.866 & 0.000 & -3.0 \\ 0.000 & 0.000 & 1.000 & 9.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

compute ${}^{A}V$.

2.13 [21] The following frame definitions are given as known:

$$ {}^{U}_{A}T = \begin{bmatrix} 0.866 & -0.500 & 0.000 & 11.0 \\ 0.500 & 0.866 & 0.000 & -1.0 \\ 0.000 & 0.000 & 1.000 & 8.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

$$ {}^{B}_{A}T = \begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.0 \\ 0.000 & 0.866 & -0.500 & 10.0 \\ 0.000 & 0.500 & 0.866 & -20.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

$$ {}^{C}_{U}T = \begin{bmatrix} 0.866 & -0.500 & 0.000 & -3.0 \\ 0.433 & 0.750 & -0.500 & -3.0 \\ 0.250 & 0.433 & 0.866 & 3.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

Draw a frame diagram (like that of Fig. 2.15) to show their arrangement qualitatively, and solve for ${}^{B}_{C}T$.

2.14 [31] Develop a general formula to obtain ${}^{A}_{B}T$, where, starting from initial coincidence, $\{B\}$ is rotated by θ about $\hat{K}$ where $\hat{K}$ passes through the point ${}^{A}P$ (not through the origin of $\{A\}$ in general).

2.15 [34] $\{A\}$ and $\{B\}$ are frames differing only in orientation. $\{B\}$ is attained as follows: starting coincident with $\{A\}$, $\{B\}$ is rotated by θ radians about unit vector $\hat{K}$—that is,

$$ {}^{A}_{B}R = {}^{A}_{B}R_{K}(\theta). $$

Show that

$$ {}^{A}_{B}R = e^{k\theta}, $$

where

$$ K = \begin{bmatrix} 0 & -k_x & ky \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}. $$

2.16 [22] A vector must be mapped through three rotation matrices:

$$ {}^{A}P = {}^{A}_{B}R\,{}^{B}_{C}R\,{}^{C}_{D}R\,{}^{D}P. $$

One choice is to first multiply the three rotation matrices together, to form ${}^{A}_{D}R$ in the expression

$$ {}^{A}P = {}^{A}_{D}R\,{}^{D}P. $$

Another choice is to transform the vector through the matrices one at a time—that is,

$$ {}^{A}P = {}^{A}_{B}R\,{}^{B}_{C}R\,{}^{C}_{D}R\,{}^{D}P, $$
$$ {}^{A}P = {}^{A}_{B}R\,{}^{B}_{C}R\,{}^{C}P, $$
$$ {}^{A}P = {}^{A}_{B}R\,{}^{B}P, $$
$$ {}^{A}P = {}^{A}P. $$

If ${}^{D}P$ is changing at 100 Hz, we would have to recalculate ${}^{A}P$ at the same rate. However, the three rotation matrices are also changing, as reported by a vision system that gives us new values for ${}^{A}_{B}R$, ${}^{B}_{C}R$, and ${}^{C}_{D}R$ at 30 Hz. What is the best way to organize the computation to minimize the calculation effort (multiplications and additions)?

2.17 [16] Another familiar set of three coordinates that can be used to describe a point in space is cylindrical coordinates. The three coordinates are defined as illustrated in Fig. 2.23. The coordinate θ gives a direction in the xy plane along which to translate radially by an amount r. Finally, z is given to specify the height above the xy plane. Compute the Cartesian coordinates of the point ${}^{A}P$ in terms of the cylindrical coordinates θ, r, and z.

2.18 [18] Another set of three coordinates that can be used to describe a point in space is spherical coordinates. The three coordinates are defined as illustrated in Fig. 2.24. The angles α and β can be thought of as describing azimuth and elevation of a ray projecting into space. The third coordinate, r, is the radial distance along that ray to the point being described. Calculate the Cartesian coordinates of the point ${}^{A}P$ in terms of the spherical coordinates α, β, and r.

2.19 [24] An object is rotated about its $\hat{X}$ axis by an amount ϕ, and then it is rotated about its new $\hat{Y}$ axis by an amount ψ. From our study of Euler angles, we know that the resulting orientation is given by

$$ R_x(\phi)R_y(\psi), $$

whereas, if the two rotations had occurred about axes of the fixed reference frame, the result would have been

$$ R_y(\psi)R_x(\phi). $$

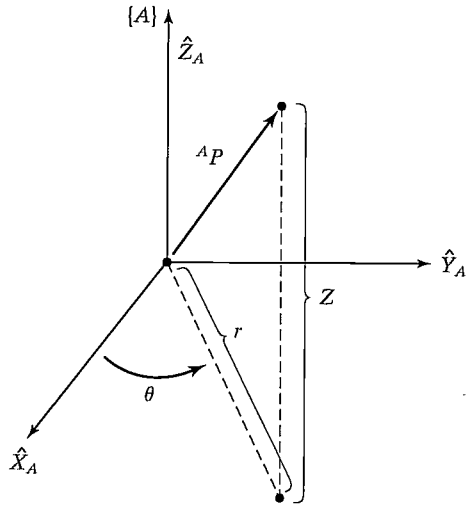

FIGURE 2.23: Cylindrical coordinates.

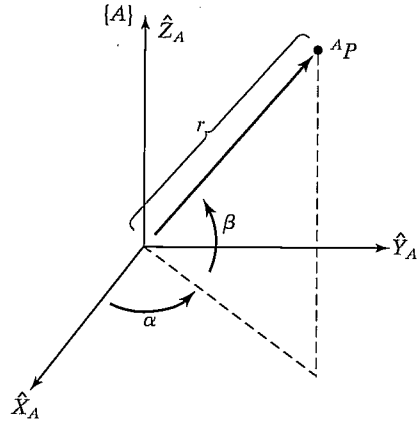

FIGURE 2.24: Spherical coordinates.

It appears that the order of multiplication depends upon whether rotations are described relative to fixed axes or those of the frame being moved. It is more appropriate, however, to realize that, in the case of specifying a rotation about an axis of the frame being moved, we are specifying a rotation in the fixed system given by (for this example)

$$R_x(\phi)R_y(\psi)R_x^{-1}(\phi).$$

This *similarity transform* [1], multiplying the original $R_x(\phi)$ on the left, reduces to the resulting expression in which *it looks as if* the order of matrix multiplication has been reversed. Taking this viewpoint, give a derivation for the form of the

rotation matrix that is equivalent to the Z–Y–Z Euler-angle set (α, β, γ). (The result is given by (2.72).)

2.20 [20] Imagine rotating a vector Q about a vector $\hat{K}$ by an amount θ to form a new vector, Q'—that is,

$$Q' = R_K(\theta)Q.$$

Use (2.80) to derive **Rodriques's formula**,

$$Q' = Q\cos\theta + \sin\theta(\hat{K} \times Q) + (1 - \cos\theta)(\hat{K} \cdot \hat{Q})\hat{K}.$$

2.21 [15] For rotations sufficiently small that the approximations $\sin\theta = \theta$, $\cos\theta = 1$, and $\theta^2 = 0$ hold, derive the rotation-matrix equivalent to a rotation of θ about a general axis, $\hat{K}$. Start with (2.80) for your derivation.

2.22 [20] Using the result from Exercise 2.21, show that two infinitesimal rotations commute (i.e., the order in which the rotations are performed is not important).

2.23 [25] Give an algorithm to construct the definition of a frame ${}^U_A T$ from three points ${}^U P_1$, ${}^U P_2$, and ${}^U P_3$, where the following is known about these points:

1. ${}^U P_1$ is at the origin of $\{A\}$;
2. ${}^U P_2$ lies somewhere on the positive $\hat{X}$ axis of $\{A\}$;
3. ${}^U P_3$ lies near the positive $\hat{Y}$ axis in the XY plane of $\{A\}$.

2.24 [45] Prove Cayley's formula for proper orthonormal matrices.

2.25 [30] Show that the eigenvalues of a rotation matrix are 1, $e^{\alpha i}$, and $e^{-\alpha i}$, where $i = \sqrt{-1}$.

2.26 [33] Prove that any Euler-angle set is sufficient to express all possible rotation matrices.

2.27 [15] Referring to Fig. 2.25, give the value of ${}^A_B T$.

2.28 [15] Referring to Fig. 2.25, give the value of ${}^A_C T$.

2.29 [15] Referring to Fig. 2.25, give the value of ${}^B_C T$.

2.30 [15] Referring to Fig. 2.25, give the value of ${}^C_A T$.

2.31 [15] Referring to Fig. 2.26, give the value of ${}^A_B T$.

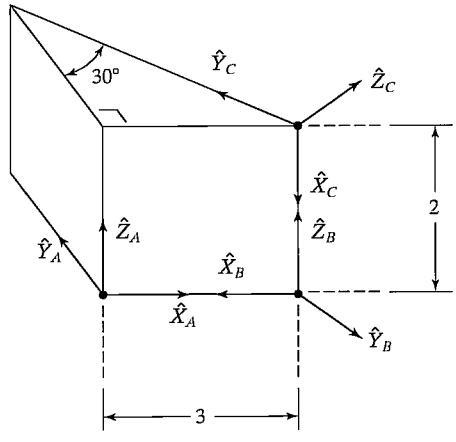

FIGURE 2.25: Frames at the corners of a wedge.

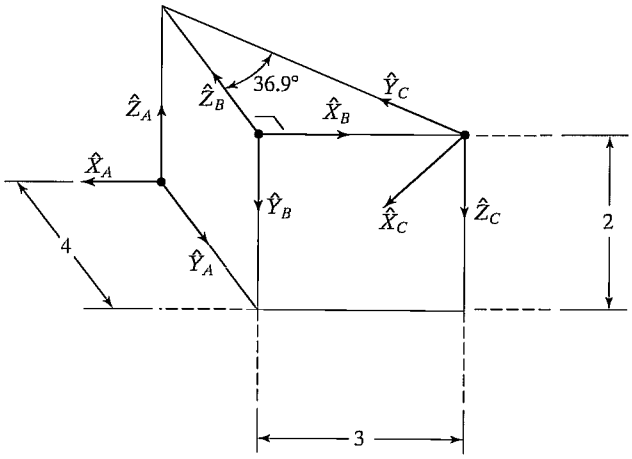

FIGURE 2.26: Frames at the corners of a wedge.

2.32 [15] Referring to Fig. 2.26, give the value of ${}^{A}_{C}T$.

2.33 [15] Referring to Fig. 2.26, give the value of ${}^{B}_{C}T$.

2.34 [15] Referring to Fig. 2.26, give the value of ${}^{C}_{A}T$.

2.35 [20] Prove that the determinant of any rotation matrix is always equal to 1.

2.36 [36] A rigid body moving in a plane (i.e., in 2-space) has three degrees of freedom. A rigid body moving in 3-space has six degrees of freedom. Show that a body in N-space has $\frac{1}{2}(N^2 + N)$ degrees of freedom.

2.37 [15] Given

$$ {}^{A}_{B}T = \begin{bmatrix} 0.25 & 0.43 & 0.86 & 5.0 \\ 0.87 & -0.50 & 0.00 & -4.0 \\ 0.43 & 0.75 & -0.50 & 3.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

what is the (2,4) element of ${}^{B}_{A}T$?

2.38 [25] Imagine two unit vectors, v_1 and v_2, embedded in a rigid body. Note that, no matter how the body is rotated, the geometric angle between these two vectors is preserved (i.e., rigid-body rotation is an "angle-preserving" operation). Use this fact to give a concise (four- or five-line) proof that the inverse of a rotation matrix must equal its transpose and that a rotation matrix is orthonormal.

2.39 [37] Give an algorithm (perhaps in the form of a C program) that computes the unit quaternion corresponding to a given rotation matrix. Use (2.91) as starting point.

2.40 [33] Give an algorithm (perhaps in the form of a C program) that computes the Z–X–Z Euler angles corresponding to a given rotation matrix. See Appendix B.

2.41 [33] Give an algorithm (perhaps in the form of a C program) that computes the X–Y–X fixed angles corresponding to a given rotation matrix. See Appendix B.

PROGRAMMING EXERCISE (PART 2)

1. If your function library does not include an Atan2 function subroutine, write one.
2. To make a friendly user interface, we wish to describe orientations in the planar world by a single angle, θ, instead of by a 2×2 rotation matrix. The user will always

communicate in terms of angle θ, but internally we will need the rotation-matrix form. For the position-vector part of a frame, the user will specify an x and a y value. So, we want to allow the user to specify a *frame* as a 3-tuple: (x, y, θ). Internally, we wish to use a 2×1 position vector and a 2×2 rotation matrix, so we need conversion routines. Write a subroutine whose Pascal definition would begin

```
Procedure UTOI (VAR uform: vec3; VAR iform: frame);
```

where "UTOI" stands for "User form TO Internal form." The first argument is the 3-tuple (x, y, θ), and the second argument is of type "frame," consists of a (2×1) position vector and a (2×2) rotation matrix. If you wish, you may represent the frame with a (3×3) homogeneous transform in which the third row is [0 0 1]. The inverse routine will also be necessary:

```
Procedure ITOU (VAR iform: frame; VAR uform: vec3);
```

3. Write a subroutine to multiply two transforms together. Use the following procedure heading:

```
Procedure TMULT (VAR brela, crelb, crela: frame);
```

The first two arguments are inputs, and the third is an output. Note that the names of the arguments document what the program does (brela $= {}^{A}_{B}T$).

4. Write a subroutine to invert a transform. Use the following procedure heading:

```
Procedure TINVERT (VAR brela, arelb: frame);
```

The first argument is the input, the second the output. Note that the names of the arguments document what the program does (brela $= {}^{A}_{B}T$).

5. The following frame definitions are given as known:

$$
\begin{aligned}
{}^{U}_{A}T &= [x\ y\ \theta] = [11.0\ -1.0\ 30.0], \\
{}^{B}_{A}T &= [x\ y\ \theta] = [0.0\ 7.0\ 45.0], \\
{}^{C}_{U}T &= [x\ y\ \theta] = [-3.0\ -3.0\ -30.0].
\end{aligned}
$$

These frames are input in the user representation $[x, y, \theta]$ (where θ is in degrees). Draw a frame diagram (like Fig. 2.15, only in 2-D) that qualitatively shows their arrangement. Write a program that calls TMULT and TINVERT (defined in programming exercises 3 and 4) as many times as needed to solve for ${}^{B}_{C}T$. Then print out ${}^{B}_{C}T$ in both internal and user representation.

MATLAB EXERCISE 2A

a) Using the Z–Y–X $(\alpha - \beta - \gamma)$ Euler angle convention, write a MATLAB program to calculate the rotation matrix ${}^{A}_{B}R$ when the user enters the Euler angles α–β–γ. Test for two examples:

i) $\alpha = 10^{\circ}, \quad \beta = 20^{\circ}, \quad \gamma = 30^{\circ}.$
ii) $\alpha = 30^{\circ}, \quad \beta = 90^{\circ}, \quad \gamma = -55^{\circ}.$

For case (i), demonstrate the six constraints for unitary orthonormal rotation matrices (i.e., there are nine numbers in a 3×3 matrix, but only three are independent). Also, demonstrate the *beautiful* property, ${}^{B}_{A}R = {}^{A}_{B}R^{-1} = {}^{A}_{B}R^{T}$, for case i.

b) Write a MATLAB program to calculate the Euler angles α–β–γ when the user enters the rotation matrix ${}^{A}_{B}R$ (the inverse problem). Calculate both possible solutions. Demonstrate this inverse solution for the two cases from part (a). Use a circular check to verify your results (i.e., enter Euler angles in code *a* from part (a); take the resulting rotation matrix ${}^{A}_{B}R$, and use this as the input to code *b*; you get two sets of answers—one should be the original user input, and the second can be verified by once again using the code in part (a).

c) For a simple rotation of β about the Y axis only, for $\beta = 20^{\circ}$ and ${}^{B}P = \{1\ \ 0\ \ 1\}^{T}$, calculate ${}^{A}P$; demonstrate with a sketch that your results are correct.

d) Check all results, by means of the Corke MATLAB Robotics Toolbox. Try the functions *rpy2tr*(), *tr2rpy*(), *rotx*(), *roty*(), and *rotz*().

MATLAB EXERCISE 2B

a) Write a MATLAB program to calculate the homogeneous transformation matrix ${}^{A}_{B}T$ when the user enters Z–Y–X Euler angles $\alpha - \beta - \gamma$ and the position vector ${}^{A}P_{B}$. Test for two examples:

i) $\alpha = 10^{\circ}$, $\beta = 20^{\circ}$, $\gamma = 30^{\circ}$, and ${}^{A}P_{B} = \{1\ \ 2\ \ 3\}^{T}$.

ii) For $\beta = 20^{\circ}$ ($\alpha = \gamma = 0^{\circ}$), ${}^{A}P_{B} = \{3\ \ 0\ \ 1\}^{T}$.

b) For $\beta = 20^{\circ}$ ($\alpha = \gamma = 0^{\circ}$), ${}^{A}P_{B} = \{3\ \ 0\ \ 1\}^{T}$, and ${}^{B}P = \{1\ \ 0\ \ 1\}^{T}$, use MATLAB to calculate ${}^{A}P$; demonstrate with a sketch that your results are correct. Also, using the same numbers, demonstrate all three interpretations of the homogeneous transformation matrix—the (b) assignment is the second interpretation, transform mapping.

c) Write a MATLAB program to calculate the inverse homogeneous transformation matrix ${}^{A}_{B}T^{-1} = {}^{B}_{A}T$, using the symbolic formula. Compare your result with a numerical MATLAB function (e.g., *inv*). Demonstrate that both methods yield correct results (i.e., ${}^{A}_{B}T\ {}^{A}_{B}T^{-1} = {}^{A}_{B}T^{-1}\ {}^{A}_{B}T = I_4$). Demonstrate this for examples (i) and (ii) from (a) above.

d) Define ${}^{A}_{B}T$ to be the result from (a)(i) and ${}^{B}_{C}T$ to be the result from (a)(ii).

i) Calculate ${}^{A}_{C}T$, and show the relationship via a transform graph. Do the same for ${}^{C}_{A}T$.

ii) Given ${}^{A}_{C}T$ and ${}^{B}_{C}T$ from (d)(i)—assume you don't know ${}^{A}_{B}T$, calculate it, and compare your result with the answer you know.

iii) Given ${}^{A}_{C}T$ and ${}^{A}_{B}T$ from (d)(i)—assume you don't know ${}^{B}_{C}T$, calculate it, and compare your result with the answer you know.

e) Check all results by means of the Corke MATLAB Robotics Toolbox. Try functions *rpy2tr*() and *transl*().