Chapter 3

Geometric Representations and
Transformations

**Steven M. LaValle**

**University of Illinois**

Available for downloading at **http://planning.cs.uiuc.edu/**

# Chapter 3

# Geometric Representations and Transformations

This chapter provides important background material that will be needed for Part II. Formulating and solving motion planning problems require defining and manipulating complicated geometric models of a system of bodies in space. Section 3.1 introduces geometric modeling, which focuses mainly on semi-algebraic modeling because it is an important part of Chapter 6. If your interest is mainly in Chapter 5, then understanding semi-algebraic models is not critical. Sections 3.2 and 3.3 describe how to transform a single body and a chain of bodies, respectively. This will enable the robot to "move." These sections are essential for understanding all of Part II and many sections beyond. It is expected that many readers will already have some or all of this background (especially Section 3.2, but it is included for completeness). Section 3.4 extends the framework for transforming chains of bodies to transforming trees of bodies, which allows modeling of complicated systems, such as humanoid robots and flexible organic molecules. Finally, Section 3.5 briefly covers transformations that do not assume each body is rigid.

## 3.1    Geometric Modeling

A wide variety of approaches and techniques for geometric modeling exist, and the particular choice usually depends on the application and the difficulty of the problem. In most cases, there are generally two alternatives: 1) a *boundary representation*, and 2) a *solid representation*. Suppose we would like to define a model of a planet. Using a boundary representation, we might write the equation of a sphere that roughly coincides with the planet's surface. Using a solid representation, we would describe the set of all points that are contained in the sphere. Both alternatives will be considered in this section.

The first step is to define the *world* $\mathcal{W}$ for which there are two possible choices: 1) a 2D world, in which $\mathcal{W} = \mathbb{R}^2$, and 2) a 3D world, in which $\mathcal{W} = \mathbb{R}^3$. These

choices should be sufficient for most problems; however, one might also want to allow more complicated worlds, such as the surface of a sphere or even a higher dimensional space. Such generalities are avoided in this book because their current applications are limited. Unless otherwise stated, the world generally contains two kinds of entities:

1. **Obstacles:** Portions of the world that are "permanently" occupied, for example, as in the walls of a building.

2. **Robots:** Bodies that are modeled geometrically and are controllable via a motion plan.

Based on the terminology, one obvious application is to model a robot that moves around in a building; however, many other possibilities exist. For example, the robot could be a flexible molecule, and the obstacles could be a folded protein. As another example, the robot could be a virtual human in a graphical simulation that involves obstacles (imagine the family of Doom-like video games).

This section presents a method for systematically constructing representations of obstacles and robots using a collection of primitives. Both obstacles and robots will be considered as (closed) subsets of $\mathcal{W}$. Let the *obstacle region* $\mathcal{O}$ denote the set of all points in $\mathcal{W}$ that lie in one or more obstacles; hence, $\mathcal{O} \subseteq \mathcal{W}$. The next step is to define a systematic way of representing $\mathcal{O}$ that has great expressive power while being computationally efficient. Robots will be defined in a similar way; however, this will be deferred until Section 3.2, where transformations of geometric bodies are defined.

### 3.1.1    Polygonal and Polyhedral Models

In this and the next subsection, a solid representation of $\mathcal{O}$ will be developed in terms of a combination of *primitives*. Each primitive $H_i$ represents a subset of $\mathcal{W}$ that is easy to represent and manipulate in a computer. A complicated obstacle region will be represented by taking finite, Boolean combinations of primitives. Using set theory, this implies that $\mathcal{O}$ can also be defined in terms of a finite number of unions, intersections, and set differences of primitives.

**Convex polygons**    First consider $\mathcal{O}$ for the case in which the obstacle region is a convex, polygonal subset of a 2D world, $\mathcal{W} = \mathbb{R}^2$. A subset $X \subset \mathbb{R}^n$ is called *convex* if and only if, for any pair of points in $X$, all points along the line segment that connects them are contained in $X$. More precisely, this means that for any $x_1, x_2 \in X$ and $\lambda \in [0, 1]$,

$$\lambda x_1 + (1 - \lambda)x_2 \in X. \qquad (3.1)$$

Thus, interpolation between $x_1$ and $x_2$ always yields points in $X$. Intuitively, $X$ contains no pockets or indentations. A set that is not convex is called *nonconvex* (as opposed to *concave*, which seems better suited for lenses).
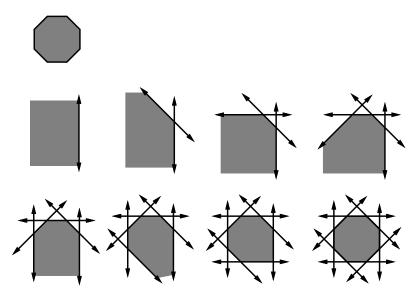
Figure 3.1: A convex polygonal region can be identified by the intersection of half-planes.

A boundary representation of $\mathcal{O}$ is an $m$-sided polygon, which can be described using two kinds of *features*: vertices and edges. Every *vertex* corresponds to a "corner" of the polygon, and every *edge* corresponds to a line segment between a pair of vertices. The polygon can be specified by a sequence, $(x_1, y_1)$, $(x_2, y_2)$, ..., $(x_m, y_m)$, of $m$ points in $\mathbb{R}^2$, given in counterclockwise order.

A solid representation of $\mathcal{O}$ can be expressed as the intersection of $m$ half-planes. Each half-plane corresponds to the set of all points that lie to one side of a line that is common to a polygon edge. Figure 3.1 shows an example of an octagon that is represented as the intersection of eight half-planes.

An edge of the polygon is specified by two points, such as $(x_1, y_1)$ and $(x_2, y_2)$. Consider the equation of a line that passes through $(x_1, y_1)$ and $(x_2, y_2)$. An equation can be determined of the form $ax + by + c = 0$, in which $a, b, c \in \mathbb{R}$ are constants that are determined from $x_1$, $y_1$, $x_2$, and $y_2$. Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ be the function given by $f(x, y) = ax + by + c$. Note that $f(x, y) < 0$ on one side of the line, and $f(x, y) > 0$ on the other. (In fact, $f$ may be interpreted as a signed Euclidean distance from $(x, y)$ to the line.) The sign of $f(x, y)$ indicates a half-plane that is bounded by the line, as depicted in Figure 3.2. Without loss of generality, assume that $f(x, y)$ is defined so that $f(x, y) < 0$ for all points to the left of the edge from $(x_1, y_1)$ to $(x_2, y_2)$ (if it is not, then multiply $f(x, y)$ by $-1$).

Let $f_i(x, y)$ denote the $f$ function derived from the line that corresponds to the edge from $(x_i, y_i)$ to $(x_{i+1}, y_{i+1})$ for $1 \leq i < m$. Let $f_m(x, y)$ denote the line equation that corresponds to the edge from $(x_m, y_m)$ to $(x_1, y_1)$. Let a *half-plane*
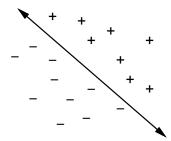
Figure 3.2: The sign of the $f(x, y)$ partitions $\mathbb{R}^2$ into three regions: two half-planes given by $f(x, y) < 0$ and $f(x, y) > 0$, and the line $f(x, y) = 0$.

$H_i$ for $1 \leq i \leq m$ be defined as a subset of $\mathcal{W}$:

$$H_i = \{(x, y) \in \mathcal{W} \mid f_i(x, y) \leq 0\}. \tag{3.2}$$

Above, $H_i$ is a primitive that describes the set of all points on one side of the line $f_i(x, y) = 0$ (including the points on the line). A convex, $m$-sided, polygonal obstacle region $\mathcal{O}$ is expressed as

$$\mathcal{O} = H_1 \cap H_2 \cap \cdots \cap H_m. \tag{3.3}$$

**Nonconvex polygons** The assumption that $\mathcal{O}$ is convex is too limited for most applications. Now suppose that $\mathcal{O}$ is a nonconvex, polygonal subset of $\mathcal{W}$. In this case $\mathcal{O}$ can be expressed as

$$\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2 \cup \cdots \cup \mathcal{O}_n, \tag{3.4}$$

in which each $\mathcal{O}_i$ is a convex, polygonal set that is expressed in terms of half-planes using (3.3). Note that $\mathcal{O}_i$ and $\mathcal{O}_j$ for $i \neq j$ need not be disjoint. Using this representation, very complicated obstacle regions in $\mathcal{W}$ can be defined. Although these regions may contain multiple components and holes, if $\mathcal{O}$ is bounded (i.e., $\mathcal{O}$ will fit inside of a big enough rectangular box), its boundary will consist of linear segments.

In general, more complicated representations of $\mathcal{O}$ can be defined in terms of any finite combination of unions, intersections, and set differences of primitives; however, it is always possible to simplify the representation into the form given by (3.3) and (3.4). A set difference can be avoided by redefining the primitive. Suppose the model requires removing a set defined by a primitive $H_i$ that contains[1] $f_i(x, y) < 0$. This is equivalent to keeping all points such that $f_i(x, y) \geq 0$, which is equivalent to $-f_i(x, y) \leq 0$. This can be used to define a new primitive $H_i'$, which

---

[1]In this section, we want the resulting set to include all of the points along the boundary. Therefore, $<$ is used to model a set for removal, as opposed to $\leq$.

when taken in union with other sets, is equivalent to the removal of $H_i$. Given a complicated combination of primitives, once set differences are removed, the expression can be simplified into a finite union of finite intersections by applying Boolean algebra laws.

Note that the representation of a nonconvex polygon is not unique. There are many ways to decompose $\mathcal{O}$ into convex components. The decomposition should be carefully selected to optimize computational performance in whatever algorithms that model will be used. In most cases, the components may even be allowed to overlap. Ideally, it seems that it would be nice to represent $\mathcal{O}$ with the minimum number of primitives, but automating such a decomposition may lead to an NP-hard problem (see Section 6.5.1 for a brief overview of NP-hardness). One efficient, practical way to decompose $\mathcal{O}$ is to apply the vertical cell decomposition algorithm, which will be presented in Section 6.2.2

**Defining a logical predicate**    What is the value of the previous representation? As a simple example, we can define a logical predicate that serves as a collision detector. Recall from Section 2.4.1 that a predicate is a Boolean-valued function. Let $\phi$ be a predicate defined as $\phi : \mathcal{W} \to \{\text{TRUE}, \text{FALSE}\}$, which returns TRUE for a point in $\mathcal{W}$ that lies in $\mathcal{O}$, and FALSE otherwise. For a line given by $f(x,y) = 0$, let $e(x,y)$ denote a logical predicate that returns TRUE if $f(x,y) \leq 0$, and FALSE otherwise.

A predicate that corresponds to a convex polygonal region is represented by a logical conjunction,

$$\alpha(x,y) = e_1(x,y) \wedge e_2(x,y) \wedge \cdots \wedge e_m(x,y). \tag{3.5}$$

The predicate $\alpha(x,y)$ returns TRUE if the point $(x,y)$ lies in the convex polygonal region, and FALSE otherwise. An obstacle region that consists of $n$ convex polygons is represented by a logical disjunction of conjuncts,

$$\phi(x,y) = \alpha_1(x,y) \vee \alpha_2(x,y) \vee \cdots \vee \alpha_n(x,y). \tag{3.6}$$

Although more efficient methods exist, $\phi$ can check whether a point $(x,y)$ lies in $\mathcal{O}$ in time $O(n)$, in which $n$ is the number of primitives that appear in the representation of $\mathcal{O}$ (each primitive is evaluated in constant time).

Note the convenient connection between a logical predicate representation and a set-theoretic representation. Using the logical predicate, the unions and intersections of the set-theoretic representation are replaced by logical ORs and ANDs. It is well known from Boolean algebra that any complicated logical sentence can be reduced to a logical disjunction of conjunctions (this is often called "sum of products" in computer engineering). This is equivalent to our previous statement that $\mathcal{O}$ can always be represented as a union of intersections of primitives.

**Polyhedral models**    For a 3D world, $\mathcal{W} = \mathbb{R}^3$, and the previous concepts can be nicely generalized from the 2D case by replacing polygons with polyhedra and

replacing half-plane primitives with half-space primitives. A boundary representation can be defined in terms of three features: vertices, edges, and faces. Every face is a "flat" polygon embedded in $\mathbb{R}^3$. Every edge forms a boundary between two faces. Every vertex forms a boundary between three or more edges.

Several data structures have been proposed that allow one to conveniently "walk" around the polyhedral features. For example, the *doubly connected edge list* [5] data structure contains three types of records: faces, half-edges, and vertices. Intuitively, a half-edge is a directed edge. Each vertex record holds the point coordinates and a pointer to an arbitrary half-edge that touches the vertex. Each face record contains a pointer to an arbitrary half-edge on its boundary. Each face is bounded by a circular list of half-edges. There is a pair of directed half-edge records for each edge of the polyhedon. Each half-edge is shown as an arrow in Figure 3.3b. Each half-edge record contains pointers to five other records: 1) the vertex from which the half-edge originates; 2) the "twin" half-edge, which bounds the neighboring face, and has the opposite direction; 3) the face that is bounded by the half-edge; 4) the next element in the circular list of edges that bound the face; and 5) the previous element in the circular list of edges that bound the face. Once all of these records have been defined, one can conveniently traverse the structure of the polyhedron.

Now consider a solid representation of a polyhedron. Suppose that $\mathcal{O}$ is a convex polyhedron, as shown in Figure 3.3. A solid representation can be constructed from the vertices. Each face of $\mathcal{O}$ has at least three vertices along its boundary. Assuming these vertices are not collinear, an equation of the plane that passes through them can be determined of the form

$$ax + by + cz + d = 0, \tag{3.7}$$

in which $a, b, c, d \in \mathbb{R}$ are constants.

Once again, $f$ can be constructed, except now $f : \mathbb{R}^3 \to \mathbb{R}$ and

$$f(x,y,z) = ax + by + cz + d. \tag{3.8}$$

Let $m$ be the number of faces. For each face of $\mathcal{O}$, a *half-space* $H_i$ is defined as a subset of $\mathcal{W}$:

$$H_i = \{(x,y,z) \in \mathcal{W} \mid f_i(x,y,z) \leq 0\}. \tag{3.9}$$

It is important to choose $f_i$ so that it takes on negative values inside of the polyhedron. In the case of a polygonal model, it was possible to consistently define $f_i$ by proceeding in counterclockwise order around the boundary. In the case of a polyhedron, the half-edge data structure can be used to obtain for each face the list of edges that form its boundary in counterclockwise order. Figure 3.3b shows the edge ordering for each face. For every edge, the arrows point in opposite directions, as required by the half-edge data structure. The equation for each face can be consistently determined as follows. Choose three consecutive vertices, $p_1$, $p_2$, $p_3$ (they must not be collinear) in counterclockwise order on the
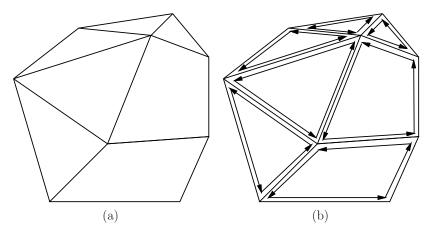
(a) (b)

Figure 3.3: (a) A polyhedron can be described in terms of faces, edges, and vertices. (b) The edges of each face can be stored in a circular list that is traversed in counterclockwise order with respect to the outward normal vector of the face.

boundary of the face. Let $v_{12}$ denote the vector from $p_1$ to $p_2$, and let $v_{23}$ denote the vector from $p_2$ to $p_3$. The cross product $v = v_{12} \times v_{23}$ always yields a vector that points out of the polyhedron and is normal to the face. Recall that the vector $[a \ \ b \ \ c]$ is parallel to the normal to the plane. If its components are chosen as $a = v[1]$, $b = v[2]$, and $c = v[3]$, then $f(x, y, z) \leq 0$ for all points in the half-space that contains the polyhedron.

As in the case of a polygonal model, a convex polyhedron can be defined as the intersection of a finite number of half-spaces, one for each face. A nonconvex polyhedron can be defined as the union of a finite number of convex polyhedra. The predicate $\phi(x, y, z)$ can be defined in a similar manner, in this case yielding TRUE if $(x, y, z) \in \mathcal{O}$, and FALSE otherwise.

## 3.1.2 Semi-Algebraic Models

In both the polygonal and polyhedral models, $f$ was a linear function. In the case of a semi-algebraic model for a 2D world, $f$ can be any polynomial with real-valued coefficients and variables $x$ and $y$. For a 3D world, $f$ is a polynomial with variables $x$, $y$, and $z$. The class of semi-algebraic models includes both polygonal and polyhedral models, which use first-degree polynomials. A point set determined by a single polynomial primitive is called an *algebraic set*; a point set that can be obtained by a finite number of unions and intersections of algebraic sets is called a *semi-algebraic set*.

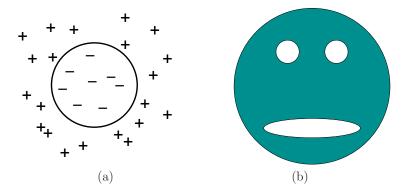Consider the case of a 2D world. A solid representation can be defined using

Figure 3.4: (a) Once again, $f$ is used to partition $\mathbb{R}^2$ into two regions. In this case, the algebraic primitive represents a disc-shaped region. (b) The shaded "face" can be exactly modeled using only four algebraic primitives.

*algebraic primitives* of the form

$$H = \{(x, y) \in \mathcal{W} \mid f(x, y) \leq 0\}. \tag{3.10}$$

As an example, let $f = x^2 + y^2 - 4$. In this case, $H$ represents a disc of radius 2 that is centered at the origin. This corresponds to the set of points $(x, y)$ for which $f(x, y) \leq 0$, as depicted in Figure 3.4a.

**Example 3.1 (Gingerbread Face)** Consider constructing a model of the shaded region shown in Figure 3.4b. Let the center of the outer circle have radius $r_1$ and be centered at the origin. Suppose that the "eyes" have radius $r_2$ and $r_3$ and are centered at $(x_2, y_2)$ and $(x_3, y_3)$, respectively. Let the "mouth" be an ellipse with major axis $a$ and minor axis $b$ and is centered at $(0, y_4)$. The functions are defined as

$$\begin{aligned} f_1 &= x^2 + y^2 - r_1^2, \\ f_2 &= -\left((x - x_2)^2 + (y - y_2)^2 - r_2^2\right), \\ f_3 &= -\left((x - x_3)^2 + (y - y_3)^2 - r_3^2\right), \\ f_4 &= -\left(x^2/a^2 + (y - y_4)^2/b^2 - 1\right). \end{aligned} \tag{3.11}$$

For $f_2$, $f_3$, and $f_4$, the familiar circle and ellipse equations were multiplied by $-1$ to yield algebraic primitives for all points outside of the circle or ellipse. The shaded region $\mathcal{O}$ is represented as

$$\mathcal{O} = H_1 \cap H_2 \cap H_3 \cap H_4. \tag{3.12}$$

∎

In the case of semi-algebraic models, the intersection of primitives does not necessarily result in a convex subset of $\mathcal{W}$. In general, however, it might be necessary to form $\mathcal{O}$ by taking unions and intersections of algebraic primitives.

A logical predicate, $\phi(x, y)$, can once again be formed, and collision checking is still performed in time that is linear in the number of primitives. Note that it is still very efficient to evaluate every primitive; $f$ is just a polynomial that is evaluated on the point $(x, y, z)$.

The semi-algebraic formulation generalizes easily to the case of a 3D world. This results in algebraic primitives of the form

$$H = \{(x, y, z) \in \mathcal{W} \mid f(x, y, z) \leq 0\}, \tag{3.13}$$

which can be used to define a solid representation of a 3D obstacle $\mathcal{O}$ and a logical predicate $\phi$.

Equations (3.10) and (3.13) are sufficient to express any model of interest. One may define many other primitives based on different relations, such as $f(x, y, z) \geq 0$, $f(x, y, z) = 0$, $f(x, y, z) < 0$, $f(x, y, z) = 0$, and $f(x, y, z) \neq 0$; however, most of them do not enhance the set of models that can be expressed. They might, however, be more convenient in certain contexts. To see that some primitives do not allow new models to be expressed, consider the primitive

$$H = \{(x, y, z) \in \mathcal{W} \mid f(x, y, z) \geq 0\}. \tag{3.14}$$

The right part may be alternatively represented as $-f(x, y, z) \leq 0$, and $-f$ may be considered as a new polynomial function of $x$, $y$, and $z$. For an example that involves the $=$ relation, consider the primitive

$$H = \{(x, y, z) \in \mathcal{W} \mid f(x, y, z) = 0\}. \tag{3.15}$$

It can instead be constructed as $H = H_1 \cap H_2$, in which

$$H_1 = \{(x, y, z) \in \mathcal{W} \mid f(x, y, z) \leq 0\} \tag{3.16}$$

and

$$H_2 = \{(x, y, z) \in \mathcal{W} \mid -f(x, y, z) \leq 0\}. \tag{3.17}$$

The relation $<$ does add some expressive power if it is used to construct primitives.[2] It is needed to construct models that do not include the outer boundary (for example, the set of all points *inside* of a sphere, which does not include points *on* the sphere). These are generally called *open sets* and are defined Chapter 4.

---

[2]An alternative that yields the same expressive power is to still use $\leq$, but allow set complements, in addition to unions and intersections.
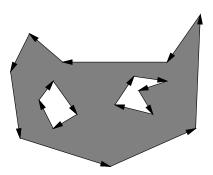
Figure 3.5: A polygon with holes can be expressed by using different orientations: counterclockwise for the outer boundary and clockwise for the hole boundaries. Note that the shaded part is always to the left when following the arrows.

### 3.1.3 Other Models

The choice of a model often depends on the types of operations that will be performed by the planning algorithm. For combinatorial motion planning methods, to be covered in Chapter 6, the particular representation is critical. On the other hand, for sampling-based planning methods, to be covered in Chapter 5, the particular representation is important only to the collision detection algorithm, which is treated as a "black box" as far as planning is concerned. Therefore, the models given in the remainder of this section are more likely to appear in sampling-based approaches and may be invisible to the designer of a planning algorithm (although it is never wise to forget completely about the representation).

**Nonconvex polygons and polyhedra** The method in Section 3.1.1 required nonconvex polygons to be represented as a union of convex polygons. Instead, a boundary representation of a nonconvex polygon may be directly encoded by listing vertices in a specific order; assume that counterclockwise order is used. Each polygon of $m$ vertices may be encoded by a list of the form $(x_1, y_1)$, $(x_2, y_2)$, ..., $(x_m, y_m)$. It is assumed that there is an edge between each $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$ for each $i$ from 1 to $m - 1$, and also an edge between $(x_m, y_m)$ and $(x_1, y_1)$. Ordinarily, the vertices should be chosen in a way that makes the polygon *simple*, meaning that no edges intersect. In this case, there is a well-defined interior of the polygon, which is to the left of every edge, if the vertices are listed in counterclockwise order.

What if a polygon has a hole in it? In this case, the boundary of the hole can be expressed as a polygon, but with its vertices appearing in the clockwise direction. To the left of each edge is the interior of the outer polygon, and to the right is the hole, as shown in Figure 3.5

Although the data structures are a little more complicated for three dimensions, boundary representations of nonconvex polyhedra may be expressed in a

Figure 3.6: Triangle strips and triangle fans can reduce the number of redundant points.

similar manner. In this case, instead of an edge list, one must specify faces, edges, and vertices, with pointers that indicate their incidence relations. Consistent orientations must also be chosen, and holes may be modeled once again by selecting opposite orientations.

**3D triangles** Suppose $\mathcal{W} = \mathbb{R}^3$. One of the most convenient geometric models to express is a set of triangles, each of which is specified by three points, $(x_1, y_1, z_1)$, $(x_2, y_2, z_2)$, $(x_3, y_3, z_3)$. This model has been popular in computer graphics because graphics acceleration hardware primarily uses triangle primitives. It is assumed that the interior of the triangle is part of the model. Thus, two triangles are considered as "colliding" if one pokes into the interior of another. This model offers great flexibility because there are no constraints on the way in which triangles must be expressed; however, this is also one of the drawbacks. There is no coherency that can be exploited to easily declare whether a point is "inside" or "outside" of a 3D obstacle. If there is at least some coherency, then it is sometimes preferable to reduce redundancy in the specification of triangle coordinates (many triangles will share the same corners). Representations that remove this redundancy are called a *triangle strip*, which is a sequence of triangles such that each adjacent pair shares a common edge, and a *triangle fan*, which is a triangle strip in which all triangles share a common vertex. See Figure 3.6.

**Nonuniform rational B-splines (NURBS)** These are used in many engineering design systems to allow convenient design and adjustment of curved surfaces, in applications such as aircraft or automobile body design. In contrast to semi-algebraic models, which are implicit equations, NURBS and other splines are parametric equations. This makes computations such as rendering easier; however, others, such as collision detection, become more difficult. These models may be defined in any dimension. A brief 2D formulation is given here.

A curve can be expressed as

$$C(u) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,k}(u)}{\sum_{i=0}^{n} w_i N_{i,k}(u)}, \tag{3.18}$$

in which $w_i \in \mathbb{R}$ are *weights* and $P_i$ are control points. The $N_{i,k}$ are normalized

basis functions of degree $k$, which can be expressed recursively as

$$N_{i,k}(u) = \left(\frac{u - t_i}{t_{i+k} - t_i}\right) N_{i,k-1}(u) + \left(\frac{t_{i+k+1} - u}{t_{i+k+1} - t_{i+1}}\right) N_{i+1,k-1}(u). \tag{3.19}$$

The basis of the recursion is $N_{i,0}(u) = 1$ if $t_i \leq u < t_{i+1}$, and $N_{i,0}(u) = 0$ otherwise. A *knot vector* is a nondecreasing sequence of real values, $\{t_0, t_1, \ldots, t_m\}$, that controls the intervals over which certain basic functions take effect.

**Bitmaps** For either $\mathcal{W} = \mathbb{R}^2$ or $\mathcal{W} = \mathbb{R}^3$, it is possible to discretize a bounded portion of the world into rectangular cells that may or may not be occupied. The resulting model looks very similar to Example 2.1. The resolution of this discretization determines the number of cells per axis and the quality of the approximation. The representation may be considered as a binary image in which each "1" in the image corresponds to a rectangular region that contains at least one point of $\mathcal{O}$, and "0" represents those that do not contain any of $\mathcal{O}$. Although bitmaps do not have the elegance of the other models, they often arise in applications. One example is a digital map constructed by a mobile robot that explores an environment with its sensors. One generalization of bitmaps is a *gray-scale map* or *occupancy grid*. In this case, a numerical value may be assigned to each cell, indicating quantities such as "the probability that an obstacle exists" or the "expected difficulty of traversing the cell." The latter interpretation is often used in terrain maps for navigating planetary rovers.

**Superquadrics** Instead of using polynomials to define $f_i$, many generalizations can be constructed. One popular primitive is a *superquadric*, which generalizes quadric surfaces. One example is a superellipsoid, which is given for $\mathcal{W} = \mathbb{R}^3$ by

$$\left(|x/a|^{n_1} + |y/b|^{n_2}\right)^{n_1/n_2} + |z/c|^{n_1} - 1 \leq 0, \tag{3.20}$$

in which $n_1 \geq 2$ and $n_2 \geq 2$. If $n_1 = n_2 = 2$, an ellipse is generated. As $n_1$ and $n_2$ increase, the superellipsoid becomes shaped like a box with rounded corners.

**Generalized cylinders** A *generalized cylinder* is a generalization of an ordinary cylinder. Instead of being limited to a line, the center axis is a continuous *spine curve*, $(x(s), y(s), z(s))$, for some parameter $s \in [0, 1]$. Instead of a constant radius, a radius function $r(s)$ is defined along the spine. The value $r(s)$ is the radius of the circle obtained as the cross section of the generalized cylinder at the point $(x(s), y(s), z(s))$. The normal to the cross-section plane is the tangent to the spine curve at $s$.

## 3.2 Rigid-Body Transformations

Any of the techniques from Section 3.1 can be used to define both the obstacle region and the robot. Let $\mathcal{O}$ refer to the *obstacle region*, which is a subset of $\mathcal{W}$.

Let $\mathcal{A}$ refer to the robot, which is a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$, matching the dimension of $\mathcal{W}$. Although $\mathcal{O}$ remains fixed in the world, $\mathcal{W}$, motion planning problems will require "moving" the robot, $\mathcal{A}$.

## 3.2.1 General Concepts

Before giving specific transformations, it will be helpful to define them in general to avoid confusion in later parts when intuitive notions might fall apart. Suppose that a rigid robot, $\mathcal{A}$, is defined as a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$. A *rigid-body transformation* is a function, $h : \mathcal{A} \to \mathcal{W}$, that maps every point of $\mathcal{A}$ into $\mathcal{W}$ with two requirements: 1) The distance between any pair of points of $\mathcal{A}$ must be preserved, and 2) the orientation of $\mathcal{A}$ must be preserved (no "mirror images").

Using standard function notation, $h(a)$ for some $a \in \mathcal{A}$ refers to the point in $\mathcal{W}$ that is "occupied" by $a$. Let

$$h(\mathcal{A}) = \{h(a) \in \mathcal{W} \mid a \in \mathcal{A}\}, \tag{3.21}$$

which is the image of $h$ and indicates all points in $\mathcal{W}$ occupied by the transformed robot.

**Transforming the robot model** Consider transforming a robot model. If $\mathcal{A}$ is expressed by naming specific points in $\mathbb{R}^2$, as in a boundary representation of a polygon, then each point is simply transformed from $a$ to $h(a) \in \mathcal{W}$. In this case, it is straightforward to transform the entire model using $h$. However, there is a slight complication if the robot model is expressed using primitives, such as

$$H_i = \{a \in \mathbb{R}^2 \mid f_i(a) \leq 0\}. \tag{3.22}$$

This differs slightly from (3.2) because the robot is defined in $\mathbb{R}^2$ (which is not necessarily $\mathcal{W}$), and also $a$ is used to denote a point $(x, y) \in \mathcal{A}$. Under a transformation $h$, the primitive is transformed as

$$h(H_i) = \{h(a) \in \mathcal{W} \mid f_i(a) \leq 0\}. \tag{3.23}$$

To transform the primitive completely, however, it is better to directly name points in $w \in \mathcal{W}$, as opposed to $h(a) \in \mathcal{W}$. Using the fact that $a = h^{-1}(w)$, this becomes

$$h(H_i) = \{w \in \mathcal{W} \mid f_i(h^{-1}(w)) \leq 0\}, \tag{3.24}$$

in which the inverse of $h$ appears in the right side because the original point $a \in A$ needs to be recovered to evaluate $f_i$. Therefore, it is important to be careful because either $h$ or $h^{-1}$ may be required to transform the model. This will be observed in more specific contexts in some coming examples.

**A parameterized family of transformations** It will become important to study families of transformations, in which some parameters are used to select the particular transformation. Therefore, it makes sense to generalize $h$ to accept two variables: a parameter vector, $q \in \mathbb{R}^n$, along with $a \in \mathcal{A}$. The resulting transformed point $a$ is denoted by $h(q, a)$, and the entire robot is transformed to $h(q, \mathcal{A}) \subset \mathcal{W}$.

The coming material will use the following shorthand notation, which requires the specific $h$ to be inferred from the context. Let $h(q, a)$ be shortened to $a(q)$, and let $h(q, \mathcal{A})$ be shortened to $\mathcal{A}(q)$. This notation makes it appear that by adjusting the parameter $q$, the robot $\mathcal{A}$ travels around in $\mathcal{W}$ as different transformations are selected from the predetermined family. This is slightly abusive notation, but it is convenient. The expression $\mathcal{A}(q)$ can be considered as a set-valued function that yields the set of points in $\mathcal{W}$ that are occupied by $\mathcal{A}$ when it is transformed by $q$. Most of the time the notation does not cause trouble, but when it does, it is helpful to remember the definitions from this section, especially when trying to determine whether $h$ or $h^{-1}$ is needed.

**Defining frames** It was assumed so far that $\mathcal{A}$ is defined in $\mathbb{R}^2$ or $\mathbb{R}^3$, but before it is transformed, it is not considered to be a subset of $\mathcal{W}$. The transformation $h$ *places* the robot in $\mathcal{W}$. In the coming material, it will be convenient to indicate this distinction using coordinate frames. The origin and coordinate basis vectors of $\mathcal{W}$ will be referred to as the *world frame*.[3] Thus, any point $w \in \mathcal{W}$ is expressed in terms of the world frame.

The coordinates used to define $\mathcal{A}$ are initially expressed in the *body frame*, which represents the origin and coordinate basis vectors of $\mathbb{R}^2$ or $\mathbb{R}^3$. In the case of $\mathcal{A} \subset \mathbb{R}^2$, it can be imagined that the body frame is painted on the robot. Transforming the robot is equivalent to converting its model from the body frame to the world frame. This has the effect of *placing*[4] $\mathcal{A}$ into $\mathcal{W}$ at some position and orientation. When multiple bodies are covered in Section 3.3, each body will have its own body frame, and transformations require expressing all bodies with respect to the world frame.

## 3.2.2 2D Transformations

**Translation** A rigid robot $\mathcal{A} \subset \mathbb{R}^2$ is *translated* by using two parameters, $x_t, y_t \in \mathbb{R}$. Using definitions from Section 3.2.1, $q = (x_t, y_t)$, and $h$ is defined as

$$h(x, y) = (x + x_t, y + y_t). \tag{3.25}$$

---

[3]The world frame serves the same purpose as an inertial frame in Newtonian mechanics. Intuitively, it is a frame that remains fixed and from which all measurements are taken. See Section 13.3.1.

[4]Technically, this placement is a function called an *orientation-preserving isometric embedding*.

A boundary representation of $\mathcal{A}$ can be translated by transforming each vertex in the sequence of polygon vertices using (3.25). Each point, $(x_i, y_i)$, in the sequence is replaced by $(x_i + x_t, y_i + y_t)$.

Now consider a solid representation of $\mathcal{A}$, defined in terms of primitives. Each primitive of the form

$$H_i = \{(x, y) \in \mathbb{R}^2 \mid f(x, y) \le 0\} \tag{3.26}$$

is transformed to

$$h(H_i) = \{(x, y) \in \mathcal{W} \mid f(x - x_t, y - y_t) \le 0\}. \tag{3.27}$$

**Example 3.2 (Translating a Disc)** For example, suppose the robot is a disc of unit radius, centered at the origin. It is modeled by a single primitive,

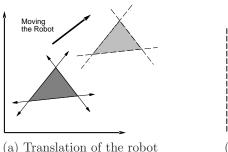$$H_i = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 - 1 \le 0\}. \tag{3.28}$$

Suppose $\mathcal{A} = H_i$ is translated $x_t$ units in the $x$ direction and $y_t$ units in the $y$ direction. The transformed primitive is

$$h(H_i) = \{(x, y) \in \mathcal{W} \mid (x - x_t)^2 + (y - y_t)^2 - 1 \le 0\}, \tag{3.29}$$

which is the familiar equation for a disc centered at $(x_t, y_t)$. In this example, the inverse, $h^{-1}$ is used, as described in Section 3.2.1. ∎

The translated robot is denoted as $\mathcal{A}(x_t, y_t)$. Translation by $(0, 0)$ is the *identity transformation*, which results in $\mathcal{A}(0, 0) = \mathcal{A}$, if it is assumed that $\mathcal{A} \subset \mathcal{W}$ (recall that $\mathcal{A}$ does not necessarily have to be initially embedded in $\mathcal{W}$). It will be convenient to use the term *degrees of freedom* to refer to the maximum number of independent parameters that are needed to completely characterize the transformation applied to the robot. If the set of allowable values for $x_t$ and $y_t$ forms a two-dimensional subset of $\mathbb{R}^2$, then the degrees of freedom is two.

Suppose that $\mathcal{A}$ is defined directly in $\mathcal{W}$ with translation. As shown in Figure 3.7, there are two interpretations of a rigid-body transformation applied to $\mathcal{A}$: 1) The world frame remains fixed and the robot is transformed; 2) the robot remains fixed and the world frame is translated. The first one characterizes the effect of the transformation from a fixed world frame, and the second one indicates how the transformation appears from the robot's perspective. Unless stated otherwise, the first interpretation will be used when we refer to motion planning problems because it often models a robot moving in a physical world. Numerous books cover coordinate transformations under the second interpretation. This has been known to cause confusion because the transformations may sometimes appear "backward" from what is desired in motion planning.
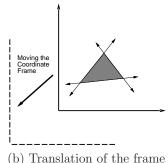
(a) Translation of the robot      (b) Translation of the frame

Figure 3.7: Every transformation has two interpretations.

**Rotation** The robot, $\mathcal{A}$, can be *rotated* counterclockwise by some angle $\theta \in [0, 2\pi)$ by mapping every $(x, y) \in \mathcal{A}$ as

$$(x, y) \mapsto (x \cos \theta - y \sin \theta, \ x \sin \theta + y \cos \theta). \tag{3.30}$$

Using a $2 \times 2$ rotation matrix,

$$R(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}, \tag{3.31}$$

the transformation can be written as

$$\begin{pmatrix} x \cos \theta - y \sin \theta \\ x \sin \theta + y \cos \theta \end{pmatrix} = R(\theta) \begin{pmatrix} x \\ y \end{pmatrix}. \tag{3.32}$$

Using the notation of Section 3.2.1, $R(\theta)$ becomes $h(q)$, for which $q = \theta$. For linear transformations, such as the one defined by (3.32), recall that the column vectors represent the basis vectors of the new coordinate frame. The column vectors of $R(\theta)$ are unit vectors, and their inner product (or dot product) is zero, indicating that they are orthogonal. Suppose that the $x$ and $y$ coordinate axes, which represent the body frame, are "painted" on $\mathcal{A}$. The columns of $R(\theta)$ can be derived by considering the resulting directions of the $x$- and $y$-axes, respectively, after performing a counterclockwise rotation by the angle $\theta$. This interpretation generalizes nicely for higher dimensional rotation matrices.

Note that the rotation is performed about the origin. Thus, when defining the model of $\mathcal{A}$, the origin should be placed at the intended axis of rotation. Using the semi-algebraic model, the entire robot model can be rotated by transforming each primitive, yielding $\mathcal{A}(\theta)$. The inverse rotation, $R(-\theta)$, must be applied to each primitive.

**Combining translation and rotation**   Suppose a rotation by $\theta$ is performed, followed by a translation by $x_t, y_t$. This can be used to place the robot in any desired position and orientation. Note that translations and rotations do not commute! If the operations are applied successively, each $(x, y) \in \mathcal{A}$ is transformed to

$$\begin{pmatrix} x \cos \theta - y \sin \theta + x_t \\ x \sin \theta + y \cos \theta + y_t \end{pmatrix}. \tag{3.33}$$

The following matrix multiplication yields the same result for the first two vector components:

$$\begin{pmatrix} \cos \theta & -\sin \theta & x_t \\ \sin \theta & \cos \theta & y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \cos \theta - y \sin \theta + x_t \\ x \sin \theta + y \cos \theta + y_t \\ 1 \end{pmatrix}. \tag{3.34}$$

This implies that the $3 \times 3$ matrix,

$$T = \begin{pmatrix} \cos \theta & -\sin \theta & x_t \\ \sin \theta & \cos \theta & y_t \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.35}$$

represents a rotation followed by a translation. The matrix $T$ will be referred to as a *homogeneous transformation matrix*. It is important to remember that $T$ represents a rotation *followed by* a translation (not the other way around). Each primitive can be transformed using the inverse of $T$, resulting in a transformed solid model of the robot. The transformed robot is denoted by $\mathcal{A}(x_t, y_t, \theta)$, and in this case there are three degrees of freedom. The homogeneous transformation matrix is a convenient representation of the combined transformations; therefore, it is frequently used in robotics, mechanics, computer graphics, and elsewhere. It is called homogeneous because over $\mathbb{R}^3$ it is just a linear transformation without any translation. The trick of increasing the dimension by one to absorb the translational part is common in projective geometry [25].

### 3.2.3   3D Transformations

Rigid-body transformations for the 3D case are conceptually similar to the 2D case; however, the 3D case appears more difficult because rotations are significantly more complicated.

**3D translation**   The robot, $\mathcal{A}$, is *translated* by some $x_t, y_t, z_t \in \mathbb{R}$ using

$$(x, y, z) \mapsto (x + x_t, y + y_t, z + z_t). \tag{3.36}$$

A primitive of the form

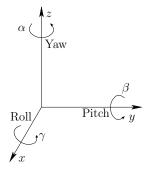$$H_i = \{(x, y, z) \in \mathcal{W} \mid f_i(x, y, z) \leq 0\} \tag{3.37}$$

Figure 3.8: Any three-dimensional rotation can be described as a sequence of yaw, pitch, and roll rotations.

is transformed to

$$\{(x, y, z) \in \mathcal{W} \mid f_i(x - x_t, y - y_t, z - z_t) \leq 0\}. \tag{3.38}$$

The translated robot is denoted as $\mathcal{A}(x_t, y_t, z_t)$.

**Yaw, pitch, and roll rotations**   A 3D body can be rotated about three orthogonal axes, as shown in Figure 3.8. Borrowing aviation terminology, these rotations will be referred to as yaw, pitch, and roll:

1. A *yaw* is a counterclockwise rotation of $\alpha$ about the $z$-axis. The rotation matrix is given by

$$R_z(\alpha) = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.39}$$

Note that the upper left entries of $R_z(\alpha)$ form a 2D rotation applied to the $x$ and $y$ coordinates, whereas the $z$ coordinate remains constant.

2. A *pitch* is a counterclockwise rotation of $\beta$ about the $y$-axis. The rotation matrix is given by

$$R_y(\beta) = \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix}. \tag{3.40}$$

3. A *roll* is a counterclockwise rotation of $\gamma$ about the $x$-axis. The rotation matrix is given by

$$R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & -\sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{pmatrix}. \tag{3.41}$$

Each rotation matrix is a simple extension of the 2D rotation matrix, (3.31). For example, the yaw matrix, $R_z(\alpha)$, essentially performs a 2D rotation with respect to the $x$ and $y$ coordinates while leaving the $z$ coordinate unchanged. Thus, the third row and third column of $R_z(\alpha)$ look like part of the identity matrix, while the upper right portion of $R_z(\alpha)$ looks like the 2D rotation matrix.

The yaw, pitch, and roll rotations can be used to place a 3D body in any orientation. A single rotation matrix can be formed by multiplying the yaw, pitch, and roll rotation matrices to obtain

$$R(\alpha,\beta,\gamma) = R_z(\alpha)\,R_y(\beta)\,R_x(\gamma) =$$

$$\begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}.$$

$$(3.42)$$

It is important to note that $R(\alpha,\beta,\gamma)$ performs the roll first, then the pitch, and finally the yaw. If the order of these operations is changed, a different rotation matrix would result. Be careful when interpreting the rotations. Consider the final rotation, a yaw by $\alpha$. Imagine sitting inside of a robot $\mathcal{A}$ that looks like an aircraft. If $\beta = \gamma = 0$, then the yaw turns the plane in a way that feels like turning a car to the left. However, for arbitrary values of $\beta$ and $\gamma$, the final rotation axis will not be vertically aligned with the aircraft because the aircraft is left in an unusual orientation before $\alpha$ is applied. The yaw rotation occurs about the $z$-axis of the world frame, not the body frame of $\mathcal{A}$. Each time a new rotation matrix is introduced from the left, it has no concern for original body frame of $\mathcal{A}$. It simply rotates every point in $\mathbb{R}^3$ in terms of the world frame. Note that 3D rotations depend on three parameters, $\alpha$, $\beta$, and $\gamma$, whereas 2D rotations depend only on a single parameter, $\theta$. The primitives of the model can be transformed using $R(\alpha,\beta,\gamma)$, resulting in $\mathcal{A}(\alpha,\beta,\gamma)$.

**Determining yaw, pitch, and roll from a rotation matrix**  It is often convenient to determine the $\alpha$, $\beta$, and $\gamma$ parameters directly from a given rotation matrix. Suppose an arbitrary rotation matrix

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

$$(3.43)$$

is given. By setting each entry equal to its corresponding entry in (3.42), equations are obtained that must be solved for $\alpha$, $\beta$, and $\gamma$. Note that $r_{21}/r_{11} = \tan\alpha$ and $r_{32}/r_{33} = \tan\gamma$. Also, $r_{31} = -\sin\beta$ and $\sqrt{r_{32}^2 + r_{33}^2} = \cos\beta$. Solving for each angle yields

$$\alpha = \tan^{-1}(r_{21}/r_{11}), \qquad (3.44)$$

$$\beta = \tan^{-1}\left(-r_{31}/\sqrt{r_{32}^2 + r_{33}^2}\right), \qquad (3.45)$$

and

$$\gamma = \tan^{-1}(r_{32}/r_{33}). \qquad (3.46)$$

There is a choice of four quadrants for the inverse tangent functions. How can the correct quadrant be determined? Each quadrant should be chosen by using the signs of the numerator and denominator of the argument. The numerator sign selects whether the direction will be above or below the $x$-axis, and the denominator selects whether the direction will be to the left or right of the $y$-axis. This is the same as the atan2 function in the C programming language, which nicely expands the range of the arctangent to $[0, 2\pi)$. This can be applied to express (3.44), (3.45), and (3.46) as

$$\alpha = \text{atan2}(r_{21}, r_{11}), \qquad (3.47)$$

$$\beta = \text{atan2}\left(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}\right), \qquad (3.48)$$

and

$$\gamma = \text{atan2}(r_{32}, r_{33}). \qquad (3.49)$$

Note that this method assumes $r_{11} \neq 0$ and $r_{33} \neq 0$.

**The homogeneous transformation matrix for 3D bodies**  As in the 2D case, a homogeneous transformation matrix can be defined. For the 3D case, a $4 \times 4$ matrix is obtained that performs the rotation given by $R(\alpha,\beta,\gamma)$, followed by a translation given by $x_t, y_t, z_t$. The result is

$$T = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & x_t \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & y_t \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & z_t \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$(3.50)$$

Once again, the order of operations is critical. The matrix $T$ in (3.50) represents the following sequence of transformations:

| | |
|---|---|
| 1. Roll by $\gamma$ | 3. Yaw by $\alpha$ |
| 2. Pitch by $\beta$ | 4. Translate by $(x_t, y_t, z_t)$. |

The robot primitives can be transformed to yield $\mathcal{A}(x_t, y_t, z_t, \alpha, \beta, \gamma)$. A 3D rigid body that is capable of translation and rotation therefore has six degrees of freedom.

## 3.3 Transforming Kinematic Chains of Bodies

The transformations become more complicated for a chain of attached rigid bodies. For convenience, each rigid body is referred to as a *link*. Let $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_m$ denote a set of $m$ links. For each $i$ such that $1 \leq i < m$, link $\mathcal{A}_i$ is "attached" to

link $\mathcal{A}_{i+1}$ in a way that allows $\mathcal{A}_{i+1}$ some constrained motion with respect to $\mathcal{A}_i$. The motion constraint must be explicitly given, and will be discussed shortly. As an example, imagine a trailer that is attached to the back of a car by a hitch that allows the trailer to rotate with respect to the car. In general, a set of attached bodies will be referred to as a *linkage*. This section considers bodies that are attached in a single chain. This leads to a particular linkage called a *kinematic chain*.

### 3.3.1   A 2D Kinematic Chain

Before considering a kinematic chain, suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are unattached rigid bodies, each of which is capable of translating and rotating in $\mathcal{W} = \mathbb{R}^2$. Since each body has three degrees of freedom, there is a combined total of six degrees of freedom; the independent parameters are $x_1$, $y_1$, $\theta_1$, $x_2$, $y_2$, and $\theta_2$.

**Attaching bodies**   When bodies are attached in a kinematic chain, degrees of freedom are removed. Figure 3.9 shows two different ways in which a pair of 2D links can be attached. The place at which the links are attached is called a *joint*. For a *revolute joint*, one link is capable only of rotation with respect to the other. For a *prismatic joint* is shown, one link slides along the other. Each type of joint removes two degrees of freedom from the pair of bodies. For example, consider a revolute joint that connects $\mathcal{A}_1$ to $\mathcal{A}_2$. Assume that the point $(0,0)$ in the body frame of $\mathcal{A}_2$ is permanently fixed to a point $(x_a, y_a)$ in the body frame of $\mathcal{A}_1$. This implies that the translation of $\mathcal{A}_2$ is completely determined once $x_a$ and $y_a$ are given. Note that $x_a$ and $y_a$ depend on $x_1$, $y_1$, and $\theta_1$. This implies that $\mathcal{A}_1$ and $\mathcal{A}_2$ have a total of four degrees of freedom when attached. The independent parameters are $x_1$, $y_1$, $\theta_1$, and $\theta_2$. The task in the remainder of this section is to determine exactly how the models of $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_m$ are transformed when they are attached in a chain, and to give the expressions in terms of the independent parameters.

Consider the case of a kinematic chain in which each pair of links is attached by a revolute joint. The first task is to specify the geometric model for each link, $\mathcal{A}_i$. Recall that for a single rigid body, the origin of the body frame determines the axis of rotation. When defining the model for a link in a kinematic chain, excessive complications can be avoided by carefully placing the body frame. Since rotation occurs about a revolute joint, a natural choice for the origin is the joint between $\mathcal{A}_i$ and $\mathcal{A}_{i-1}$ for each $i > 1$. For convenience that will soon become evident, the $x_i$-axis for the body frame of $\mathcal{A}_i$ is defined as the line through the two joints that lie in $\mathcal{A}_i$, as shown in Figure 3.10. For the last link, $\mathcal{A}_m$, the $x_m$-axis can be placed arbitrarily, assuming that the origin is placed at the joint that connects $\mathcal{A}_m$ to $\mathcal{A}_{m-1}$. The body frame for the first link, $\mathcal{A}_1$, can be placed using the same considerations as for a single rigid body.
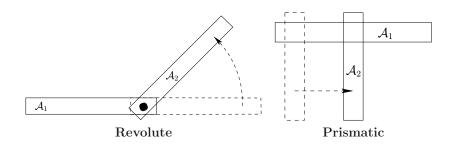
Figure 3.9: Two types of 2D joints: a revolute joint allows one link to rotate with respect to the other, and a prismatic joint allows one link to translate with respect to the other.
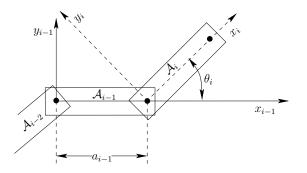


Figure 3.10: The body frame of each $\mathcal{A}_i$, for $1 < i < m$, is based on the joints that connect $\mathcal{A}_i$ to $\mathcal{A}_{i-1}$ and $\mathcal{A}_{i+1}$.

**Homogeneous transformation matrices for 2D chains** We are now prepared to determine the location of each link. The location in $\mathcal{W}$ of a point in $(x, y) \in \mathcal{A}_1$ is determined by applying the 2D homogeneous transformation matrix (3.35),

$$T_1 = \begin{pmatrix} \cos \theta_1 & -\sin \theta_1 & x_t \\ \sin \theta_1 & \cos \theta_1 & y_t \\ 0 & 0 & 1 \end{pmatrix}. \qquad (3.51)$$

As shown in Figure 3.10, let $a_{i-1}$ be the distance between the joints in $\mathcal{A}_{i-1}$. The orientation difference between $\mathcal{A}_i$ and $\mathcal{A}_{i-1}$ is denoted by the angle $\theta_i$. Let $T_i$ represent a $3 \times 3$ homogeneous transformation matrix (3.35), specialized for link $\mathcal{A}_i$ for $1 < i \leq m$,

$$T_i = \begin{pmatrix} \cos \theta_i & -\sin \theta_i & a_{i-1} \\ \sin \theta_i & \cos \theta_i & 0 \\ 0 & 0 & 1 \end{pmatrix}. \qquad (3.52)$$

This generates the following sequence of transformations:

1. Rotate counterclockwise by $\theta_i$.

2. Translate by $a_{i-1}$ along the $x$-axis.

The transformation $T_i$ expresses the difference between the body frame of $\mathcal{A}_i$ and the body frame of $\mathcal{A}_{i-1}$. The application of $T_i$ moves $\mathcal{A}_i$ from its body frame to the body frame of $\mathcal{A}_{i-1}$. The application of $T_{i-1}T_i$ moves both $\mathcal{A}_i$ and $\mathcal{A}_{i-1}$ to the body frame of $\mathcal{A}_{i-2}$. By following this procedure, the location in $\mathcal{W}$ of any point $(x, y) \in \mathcal{A}_m$ is determined by multiplying the transformation matrices to obtain

$$T_1 T_2 \cdots T_m \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \qquad (3.53)$$

**Example 3.3 (A 2D Chain of Three Links)** To gain an intuitive understanding of these transformations, consider determining the configuration for link $\mathcal{A}_3$, as shown in Figure 3.11. Figure 3.11a shows a three-link chain in which $\mathcal{A}_1$ is at its initial configuration and the other links are each offset by $\pi/4$ from the previous link. Figure 3.11b shows the frame in which the model for $\mathcal{A}_3$ is initially defined. The application of $T_3$ causes a rotation of $\theta_3$ and a translation by $a_2$. As shown in Figure 3.11c, this places $\mathcal{A}_3$ in its appropriate configuration. Note that $\mathcal{A}_2$ can be placed in its initial configuration, and it will be attached correctly to $\mathcal{A}_3$. The application of $T_2$ to the previous result places both $\mathcal{A}_3$ and $\mathcal{A}_2$ in their proper configurations, and $\mathcal{A}_1$ can be placed in its initial configuration. ∎

For revolute joints, the $a_i$ parameters are constants, and the $\theta_i$ parameters are variables. The transformed $m$th link is represented as $\mathcal{A}_m(x_t, y_t, \theta_1, \ldots, \theta_m)$. In some cases, the first link might have a fixed location in the world. In this case, the
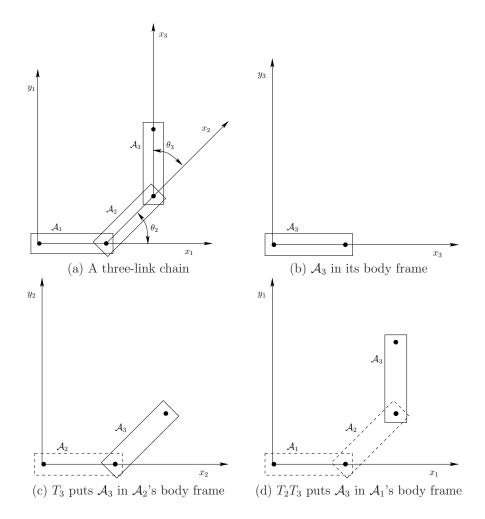


(a) A three-link chain

(b) $\mathcal{A}_3$ in its body frame

(c) $T_3$ puts $\mathcal{A}_3$ in $\mathcal{A}_2$'s body frame

(d) $T_2 T_3$ puts $\mathcal{A}_3$ in $\mathcal{A}_1$'s body frame

Figure 3.11: Applying the transformation $T_2 T_3$ to the model of $\mathcal{A}_3$. If $T_1$ is the identity matrix, then this yields the location in $\mathcal{W}$ of points in $\mathcal{A}_3$.

**Revolute**
1 Degree of Freedom

**Prismatic**
1 Degree of Freedom

**Screw**
1 Degree of Freedom

**Cylindrical**
2 Degrees of Freedom

**Spherical**
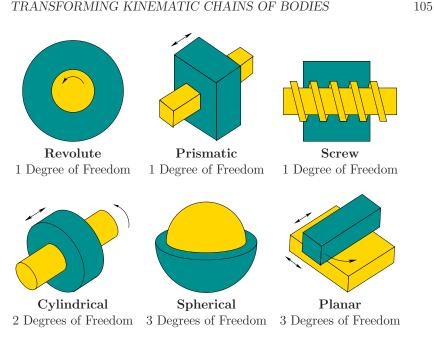3 Degrees of Freedom

**Planar**
3 Degrees of Freedom

Figure 3.12: Types of 3D joints arising from the 2D surface contact between two bodies.

revolute joints account for all degrees of freedom, yielding $\mathcal{A}_m(\theta_1, \ldots, \theta_m)$. For prismatic joints, the $a_i$ parameters are variables, instead of the $\theta_i$ parameters. It is straightforward to include both types of joints in the same kinematic chain.

### 3.3.2 A 3D Kinematic Chain

As for a single rigid body, the 3D case is significantly more complicated than the 2D case due to 3D rotations. Also, several more types of joints are possible, as shown in Figure 3.12. Nevertheless, the main ideas from the transformations of 2D kinematic chains extend to the 3D case. The following steps from Section 3.3.1 will be recycled here:

1. The body frame must be carefully placed for each $\mathcal{A}_i$.

2. Based on joint relationships, several parameters are measured.

3. The parameters define a homogeneous transformation matrix, $T_i$.

4. The location in $\mathcal{W}$ of any point in $\mathcal{A}_m$ is given by applying the matrix $T_1 T_2 \cdots T_m$.
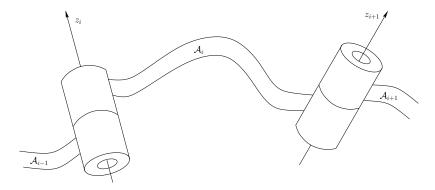


Figure 3.13: The rotation axes for a generic link attached by revolute joints.

Consider a kinematic chain of $m$ links in $\mathcal{W} = \mathbb{R}^3$, in which each $\mathcal{A}_i$ for $1 \leq i < m$ is attached to $\mathcal{A}_{i+1}$ by a revolute joint. Each link can be a complicated, rigid body as shown in Figure 3.13. For the 2D problem, the coordinate frames were based on the points of attachment. For the 3D problem, it is convenient to use the axis of rotation of each revolute joint (this is equivalent to the point of attachment for the 2D case). The axes of rotation will generally be skew lines in $\mathbb{R}^3$, as shown in Figure 3.14. Let the $z_i$-axis be the axis of rotation for the revolute joint that holds $\mathcal{A}_i$ to $\mathcal{A}_{i-1}$. Between each pair of axes in succession, let the $x_i$-axis join the closest pair of points between the $z_i$- and $z_{i+1}$-axes, with the origin on the $z_i$-axis and the direction pointing towards the nearest point of the $z_{i+1}$-axis. This axis is uniquely defined if the $z_i$- and $z_{i+1}$-axes are not parallel. The recommended body frame for each $\mathcal{A}_i$ will be given with respect to the $z_i$- and $x_i$-axes, which are shown in Figure 3.14. Assuming a right-handed coordinate system, the $y_i$-axis points away from us in Figure 3.14. In the transformations that will appear shortly, the coordinate frame given by $x_i$, $y_i$, and $z_i$ will be most convenient for defining the model for $\mathcal{A}_i$. It might not always appear convenient because the origin of the frame may even lie outside of $\mathcal{A}_i$, but the resulting transformation matrices will be easy to understand.

In Section 3.3.1, each $T_i$ was defined in terms of two parameters, $a_{i-1}$ and $\theta_i$. For the 3D case, four parameters will be defined: $d_i$, $\theta_i$, $a_{i-1}$, and $\alpha_{i-1}$. These are referred to as *Denavit-Hartenberg (DH) parameters* [9]. The definition of each parameter is indicated in Figure 3.15. Figure 3.15a shows the definition of $d_i$. Note that the $x_{i-1}$- and $x_i$-axes contact the $z_i$-axis at two different places. Let $d_i$ denote signed distance between these points of contact. If the $x_i$-axis is above the $x_{i-1}$-axis along the $z_i$-axis, then $d_i$ is positive; otherwise, $d_i$ is negative. The parameter $\theta_i$ is the angle between the $x_i$- and $x_{i-1}$-axes, which corresponds to the rotation about the $z_i$-axis that moves the $x_{i-1}$-axis to coincide with the $x_i$-axis. The parameter $a_i$ is the distance between the $z_i$- and $z_{i-1}$-axes; recall these are generally skew lines in $\mathbb{R}^3$. The parameter $\alpha_{i-1}$ is the angle between the $z_i$- and
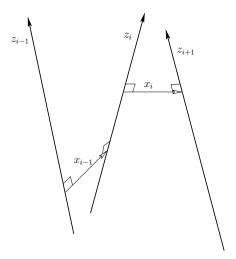
Figure 3.14: The rotation axes of the generic links are skew lines in $\mathbb{R}^3$.

$z_{i-1}$-axes.

**Two screws** The homogeneous transformation matrix $T_i$ will be constructed by combining two simpler transformations. The transformation

$$R_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.54)$$

causes a rotation of $\theta_i$ about the $z_i$-axis, and a translation of $d_i$ along the $z_i$-axis. Notice that the rotation by $\theta_i$ and translation by $d_i$ commute because both operations occur with respect to the same axis, $z_i$. The combined operation of a translation and rotation with respect to the same axis is referred to as a *screw* (as in the motion of a screw through a nut). The effect of $R_i$ can thus be considered as a screw about the $z_i$-axis. The second transformation is

$$Q_{i-1} = \begin{pmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & \cos\alpha_{i-1} & -\sin\alpha_{i-1} & 0 \\ 0 & \sin\alpha_{i-1} & \cos\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad (3.55)$$

which can be considered as a screw about the $x_{i-1}$-axis. A rotation of $\alpha_{i-1}$ about the $x_{i-1}$-axis and a translation of $a_{i-1}$ are performed.
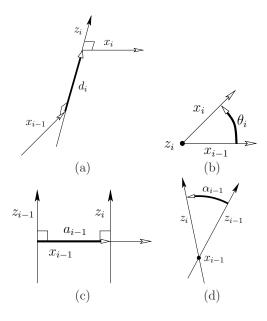
Figure 3.15: Definitions of the four DH parameters: $d_i$, $\theta_i$, $a_{i-1}$, $\alpha_{i-1}$. The $z_i$- and $x_{i-1}$-axes in (b) and (d), respectively, are pointing outward. Any parameter may be positive, zero, or negative.

**The homogeneous transformation matrix** The transformation $T_i$, for each $i$ such that $1 < i \le m$, is

$$T_i = Q_{i-1}R_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$(3.56)$$

This can be considered as the 3D counterpart to the 2D transformation matrix, (3.52). The following four operations are performed in succession:

1. Translate by $d_i$ along the $z_i$-axis.

2. Rotate counterclockwise by $\theta_i$ about the $z_i$-axis.

3. Translate by $a_{i-1}$ along the $x_{i-1}$-axis.

4. Rotate counterclockwise by $\alpha_{i-1}$ about the $x_{i-1}$-axis.

As in the 2D case, the first matrix, $T_1$, is special. To represent any position and orientation of $\mathcal{A}_1$, it could be defined as a general rigid-body homogeneous
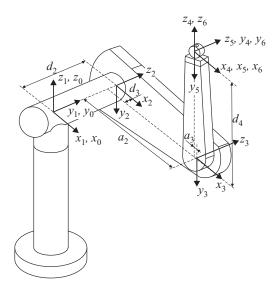
Figure 3.16: The Puma 560 is shown along with the DH parameters and body frames for each link in the chain. This figure is borrowed from [15] by courtesy of the authors.

transformation matrix, (3.50). If the first body is only capable of rotation via a revolute joint, then a simple convention is usually followed. Let the $a_0, \alpha_0$ parameters of $T_1$ be assigned as $a_0 = \alpha_0 = 0$ (there is no $z_0$-axis). This implies that $Q_0$ from (3.55) is the identity matrix, which makes $T_1 = R_1$.

The transformation $T_i$ for $i > 1$ gives the relationship between the body frame of $\mathcal{A}_i$ and the body frame of $\mathcal{A}_{i-1}$. The position of a point $(x, y, z)$ on $\mathcal{A}_m$ is given by

$$T_1 T_2 \cdots T_m \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{3.57}$$

For each revolute joint, $\theta_i$ is treated as the only variable in $T_i$. Prismatic joints can be modeled by allowing $a_i$ to vary. More complicated joints can be modeled as a sequence of degenerate joints. For example, a spherical joint can be considered as a sequence of three zero-length revolute joints; the joints perform a roll, a pitch, and a yaw. Another option for more complicated joints is to abandon the DH representation and directly develop the homogeneous transformation matrix. This might be needed to preserve topological properties that become important in Chapter 4.

**Example 3.4 (Puma 560)** This example demonstrates the 3D chain kinematics on a classic robot manipulator, the PUMA 560, shown in Figure 3.16. The

| Matrix | $\alpha_{i-1}$ | $a_{i-1}$ | $\theta_i$ | $d_i$ |
|--------|------------|-----------|-----------|-------|
| $T_1(\theta_1)$ | 0 | 0 | $\theta_1$ | 0 |
| $T_2(\theta_2)$ | $-\pi/2$ | 0 | $\theta_2$ | $d_2$ |
| $T_3(\theta_3)$ | 0 | $a_2$ | $\theta_3$ | $d_3$ |
| $T_4(\theta_4)$ | $\pi/2$ | $a_3$ | $\theta_4$ | $d_4$ |
| $T_5(\theta_5)$ | $-\pi/2$ | 0 | $\theta_5$ | 0 |
| $T_6(\theta_6)$ | $\pi/2$ | 0 | $\theta_6$ | 0 |

Figure 3.17: The DH parameters are shown for substitution into each homogeneous transformation matrix (3.56). Note that $a_3$ and $d_3$ are negative in this example (they are signed displacements, not distances).

current parameterization here is based on [2, 15]. The procedure is to determine appropriate body frames to represent each of the links. The first three links allow the hand (called an end-effector) to make large movements in $\mathcal{W}$, and the last three enable the hand to achieve a desired orientation. There are six degrees of freedom, each of which arises from a revolute joint. The body frames are shown in Figure 3.16, and the corresponding DH parameters are given in Figure 3.17. Each transformation matrix $T_i$ is a function of $\theta_i$; hence, it is written $T_i(\theta_i)$. The other parameters are fixed for this example. Only $\theta_1$, $\theta_2$, $\ldots$, $\theta_6$ are allowed to vary.

The parameters from Figure 3.17 may be substituted into the homogeneous transformation matrices to obtain

$$T_1(\theta_1) = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.58}$$

$$T_2(\theta_2) = \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -\sin\theta_2 & -\cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.59}$$

$$T_3(\theta_3) = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.60}$$

$$T_4(\theta_4) = \begin{pmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & a_3 \\ 0 & 0 & -1 & -d_4 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.61}$$
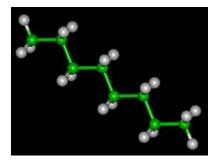
Figure 3.18: A hydrocarbon (octane) molecule with 8 carbon atoms and 18 hydrogen atoms (courtesy of the New York University MathMol Library).

$$T_5(\theta_5) = \begin{pmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_5 & -\cos\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.62}$$

and

$$T_6(\theta_6) = \begin{pmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_6 & \cos\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.63}$$

A point $(x, y, z)$ in the body frame of the last link $\mathcal{A}_6$ appears in $\mathcal{W}$ as

$$T_1(\theta_1)T_2(\theta_2)T_3(\theta_3)T_4(\theta_4)T_5(\theta_5)T_6(\theta_6) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{3.64}$$

∎

**Example 3.5 (Transforming Octane)** Figure 3.18 shows a ball-and-stick model of an octane molecule. Each "ball" is an atom, and each "stick" represents a bond between a pair of atoms. There is a linear chain of eight carbon atoms, and a bond exists between each consecutive pair of carbons in the chain. There are also numerous hydrogen atoms, but we will ignore them. Each bond between a pair of carbons is capable of twisting, as shown in Figure 3.19. Studying the configurations (called *conformations*) of molecules is an important part of computational biology. It is assumed that there are seven degrees of freedom, each of which
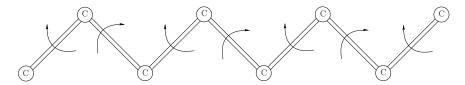
Figure 3.19: Consider transforming the spine of octane by ignoring the hydrogen atoms and allowing the bonds between carbons to rotate. This can be easily constructed with balls and sticks (e.g., Tinkertoys). If the first link is held fixed, then there are six degrees of freedom. The rotation of the last link is ignored.

arises from twisting a bond. The techniques from this section can be applied to represent these transformations.

Note that the bonds correspond exactly to the axes of rotation. This suggests that the $z_i$ axes should be chosen to coincide with the bonds. Since consecutive bonds meet at atoms, there is no distance between them. From Figure 3.15c, observe that this makes $a_i = 0$ for all $i$. From Figure 3.15a, it can be seen that each $d_i$ corresponds to a *bond length*, the distance between consecutive carbon atoms. See Figure 3.20. This leaves two angular parameters, $\theta_i$ and $\alpha_i$. Since the only possible motion of the links is via rotation of the $z_i$-axes, the angle between two consecutive axes, as shown in Figure 3.15d, must remain constant. In chemistry, this is referred to as the *bond angle* and is represented in the DH parameterization as $\alpha_i$. The remaining $\theta_i$ parameters are the variables that represent the degrees of freedom. However, looking at Figure 3.15b, observe that the example is degenerate because each $x_i$-axis has no frame of reference because each $a_i = 0$. This does not, however, cause any problems. For visualization purposes, it may be helpful to replace $x_{i-1}$ and $x_i$ by $z_{i-1}$ and $z_{i+1}$, respectively. This way it is easy to see that as the bond for the $z_i$-axis is twisted, the observed angle changes accordingly. Each bond is interpreted as a link, $\mathcal{A}_i$. The origin of each $\mathcal{A}_i$ must be chosen to coincide with the intersection point of the $z_i$- and $z_{i+1}$-axes. Thus, most of the points in $\mathcal{A}_i$ will lie in the $-z_i$ direction; see Figure 3.20.

The next task is to write down the matrices. Attach a world frame to the first bond, with the second atom at the origin and the bond aligned with the $z$-axis, in the negative direction; see Figure 3.20. To define $T_1$, recall that $T_1 = R_1$ from (3.54) because $Q_0$ is dropped. The parameter $d_1$ represents the distance between the intersection points of the $x_0$- and $x_1$-axes along the $z_1$ axis. Since there is no $x_0$-axis, there is freedom to choose $d_1$; hence, let $d_1 = 0$ to obtain

$$T_1(\theta_1) = R_1(\theta_1) = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.65}$$
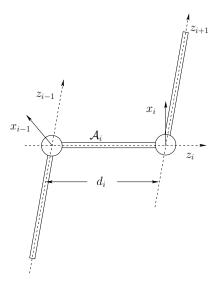
Figure 3.20: Each bond may be interpreted as a "link" of length $d_i$ that is aligned with the $z_i$-axis. Note that most of $\mathcal{A}_i$ appears in the $-z_i$ direction.

The application of $T_1$ to points in $\mathcal{A}_1$ causes them to rotate around the $z_1$-axis, which appears correct.

The matrices for the remaining six bonds are

$$T_i(\theta_i) = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (3.66)$$

for $i \in \{2, \ldots, 7\}$. The position of any point, $(x, y, z) \in \mathcal{A}_7$, is given by

$$T_1(\theta_1)T_2(\theta_2)T_3(\theta_3)T_4(\theta_4)T_5(\theta_5)T_6(\theta_6)T_7(\theta_7)\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \quad (3.67)$$

■

## 3.4 Transforming Kinematic Trees

**Motivation** For many interesting problems, the linkage is arranged in a "tree" as shown in Figure 3.21a. Assume here that the links are not attached in ways that
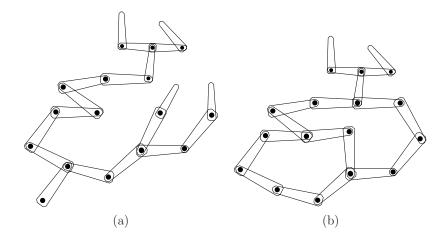
(a)          (b)

Figure 3.21: General linkages: (a) Instead of a chain of rigid bodies, a "tree" of rigid bodies can be considered. (b) If there are loops, then parameters must be carefully assigned to ensure that the loops are closed.

form loops (i.e., Figure 3.21b); that case is deferred until Section 4.4, although some comments are also made at the end of this section. The human body, with its joints and limbs attached to the torso, is an example that can be modeled as a tree of rigid links. Joints such as knees and elbows are considered as revolute joints. A shoulder joint is an example of a spherical joint, although it cannot achieve any orientation (without a visit to the emergency room!). As mentioned in Section 1.4, there is widespread interest in animating humans in virtual environments and also in developing humanoid robots. Both of these cases rely on formulations of kinematics that mimic the human body.

Another problem that involves kinematic trees is the conformational analysis of molecules. Example 3.5 involved a single chain; however, most organic molecules are more complicated, as in the familiar drugs shown in Figure 1.14a (Section 1.2). The bonds may twist to give degrees of freedom to the molecule. Moving through the space of conformations requires the formulation of a kinematic tree. Studying these conformations is important because scientists need to determine for some candidate drug whether the molecule can twist the right way so that it docks nicely (i.e., requires low energy) with a protein cavity; this induces a pharmacological effect, which hopefully is the desired one. Another important problem is determining how complicated protein molecules fold into certain configurations. These molecules are orders of magnitude larger (in terms of numbers of atoms and degrees of freedom) than typical drug molecules. For more information, see Section 7.5.
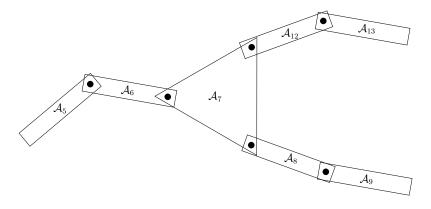
Figure 3.22: Now it is possible for a link to have more than two joints, as in $\mathcal{A}_7$.

**Common joints for $\mathcal{W} = \mathbb{R}^2$** First consider the simplest case in which there is a 2D tree of links for which every link has only two points at which revolute joints may be attached. This corresponds to Figure 3.21a. A single link is designated as the *root*, $\mathcal{A}_1$, of the tree. To determine the transformation of a body, $\mathcal{A}_i$, in the tree, the tools from Section 3.3.1 are directly applied to the chain of bodies that connects $\mathcal{A}_i$ to $\mathcal{A}_1$ while ignoring all other bodies. Each link contributes a $\theta_i$ to the total degrees of freedom of the tree. This case seems quite straightforward; unfortunately, it is not this easy in general.

**Junctions with more than two rotation axes** Now consider modeling a more complicated collection of attached links. The main novelty is that one link may have joints attached to it in more than two locations, as in $\mathcal{A}_7$ in Figure 3.22. A link with more than two joints will be referred to as a *junction*.

If there is only one junction, then most of the complications arising from junctions can be avoided by choosing the junction as the root. For example, for a simple humanoid model, the torso would be a junction. It would be sensible to make this the root of the tree, as opposed to the right foot. The legs, arms, and head could all be modeled as independent chains. In each chain, the only concern is that the first link of each chain does not attach to the same point on the torso. This can be solved by inserting a fixed, fictitious link that connects from the origin of the torso to the attachment point of the limb.

The situation is more interesting if there are multiple junctions. Suppose that Figure 3.22 represents part of a 2D system of links for which the root, $\mathcal{A}_1$, is attached via a chain of links to $\mathcal{A}_5$. To transform link $\mathcal{A}_9$, the tools from Section
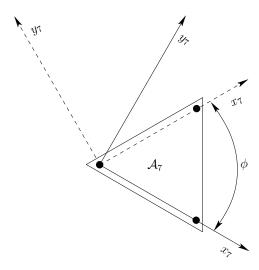
Figure 3.23: The junction is assigned two different frames, depending on which chain was followed. The solid axes were obtained from transforming $\mathcal{A}_9$, and the dashed axes were obtained from transforming $\mathcal{A}_{13}$.

3.3.1 may be directly applied to yield a sequence of transformations,

$$T_1 \cdots T_5 T_6 T_7 T_8 T_9 \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{3.68}$$

for a point $(x, y) \in \mathcal{A}_9$. Likewise, to transform $T_{13}$, the sequence

$$T_1 \cdots T_5 T_6 T_7 T_{12} T_{13} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.69}$$

can be used by ignoring the chain formed by $\mathcal{A}_8$ and $\mathcal{A}_9$. So far everything seems to work well, but take a close look at $\mathcal{A}_7$. As shown in Figure 3.23, its body frame was defined in two different ways, one for each chain. If both are forced to use the same frame, then at least one must abandon the nice conventions of Section 3.3.1 for choosing frames. This situation becomes worse for 3D trees because this would suggest abandoning the DH parameterization. The Khalil-Kleinfinger parameterization is an elegant extension of the DH parameterization and solves these frame assignment issues [13].

**Constraining parameters** Fortunately, it is fine to use different frames when following different chains; however, one extra piece of information is needed. Imag-

ine transforming the whole tree. The variable $\theta_7$ will appear twice, once from each of the upper and lower chains. Let $\theta_{7u}$ and $\theta_{7l}$ denote these $\theta$'s. Can $\theta$ really be chosen two different ways? This would imply that the tree is instead as pictured in Figure 3.24, in which there are two independently moving links, $\mathcal{A}_{7u}$ and $\mathcal{A}_{7l}$. To fix this problem, a constraint must be imposed. Suppose that $\theta_{7l}$ is treated as
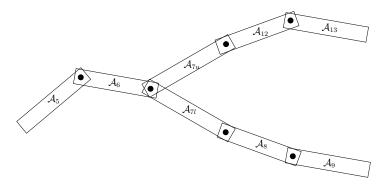


Figure 3.24: Choosing each $\theta_7$ independently would result in a tree that ignores that fact that $\mathcal{A}_7$ is rigid.

an independent variable. The parameter $\theta_{7u}$ must then be chosen as $\theta_{7l} + \phi$, in which $\phi$ is as shown in Figure 3.23.

**Example 3.6 (A 2D Tree of Bodies)** Figure 3.25 shows a 2D example that involves six links. To transform $(x, y) \in \mathcal{A}_6$, the only relevant links are $\mathcal{A}_5$, $\mathcal{A}_2$, and $\mathcal{A}_1$. The chain of transformations is

$$T_1 T_{2l} T_5 T_6 \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{3.70}$$

in which

$$T_1 = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & x_t \\ \sin\theta_1 & \cos\theta_1 & y_t \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.71}$$

$$T_{2l} = \begin{pmatrix} \cos\theta_{2l} & -\sin\theta_{2l} & a_1 \\ \sin\theta_{2l} & \cos\theta_{2l} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 1 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.72}$$

$$T_5 = \begin{pmatrix} \cos\theta_5 & -\sin\theta_5 & a_2 \\ \sin\theta_5 & \cos\theta_5 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_5 & -\sin\theta_5 & \sqrt{2} \\ \sin\theta_5 & \cos\theta_5 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.73}$$
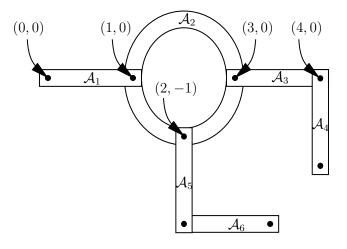
Figure 3.25: A tree of bodies in which the joints are attached in different places.

and

$$T_6 = \begin{pmatrix} \cos\theta_6 & -\sin\theta_6 & a_5 \\ \sin\theta_6 & \cos\theta_6 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_6 & -\sin\theta_6 & 1 \\ \sin\theta_6 & \cos\theta_6 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.74}$$

The matrix $T_{2l}$ in (3.72) denotes the fact that the lower chain was followed. The transformation for points in $\mathcal{A}_4$ is

$$T_1 T_{2u} T_4 T_5 \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{3.75}$$

in which $T_1$ is the same as in (3.71), and

$$T_3 = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & a_2 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & \sqrt{2} \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.76}$$

and

$$T_4 = \begin{pmatrix} \cos\theta_4 & -\sin\theta_4 & a_4 \\ \sin\theta_4 & \cos\theta_4 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_4 & -\sin\theta_4 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.77}$$

The interesting case is

$$T_{2u} = \begin{pmatrix} \cos\theta_{2u} & -\sin\theta_{2u} & a_1 \\ \sin\theta_{2u} & \cos\theta_{2u} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta_{2l} + \pi/4) & -\sin(\theta_{2l} + \pi/4) & a_1 \\ \sin(\theta_{2l} + \pi/4) & \cos(\theta_{2l} + \pi/4) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.78}$$
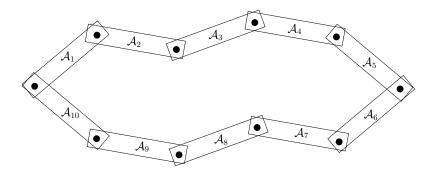
Figure 3.26: There are ten links and ten revolute joints arranged in a loop. This is an example of a closed kinematic chain.

in which the constraint $\theta_{2u} = \theta_{2l} + \pi/4$ is imposed to enforce the fact that $\mathcal{A}_2$ is a junction. ∎

For a 3D tree of bodies the same general principles may be followed. In some cases, there will not be any complications that involve special considerations of junctions and constraints. One example of this is the transformation of flexible molecules because all consecutive rotation axes intersect, and junctions occur directly at these points of intersection. In general, however, the DH parameter technique may be applied for each chain, and then the appropriate constraints have to be determined and applied to represent the true degrees of freedom of the tree. The Khalil-Kleinfinger parameterization conveniently captures the resulting solution [13].

**What if there are loops?** The most general case includes links that are connected in loops, as shown in Figure 3.26. These are generally referred to as *closed kinematic chains*. This arises in many applications. For example, with humanoid robotics or digital actors, a loop is formed when both feet touch the ground. As another example, suppose that two robot manipulators, such as the Puma 560 from Example 3.4, cooperate together to carry an object. If each robot grasps the same object with its hand, then a loop will be formed. A complicated example of this was shown in Figure 1.5, in which mobile robots moved a piano. Outside of robotics, a large fraction of organic molecules have flexible loops. Exploring the space of their conformations requires careful consideration of the difficulties imposed by these loops.

The main difficulty of working with closed kinematic chains is that it is hard to determine which parameter values are within an acceptable range to ensure closure. If these values are given, then the transformations are handled in the
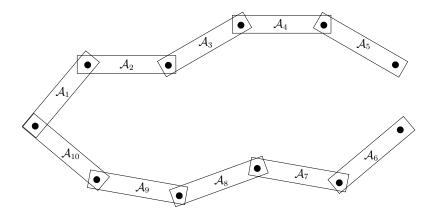
Figure 3.27: Loops may be opened to enable tree-based transformations to be applied; however, a closure constraint must still be satisfied.

same way as the case of trees. For example, the links in Figure 3.26 may be transformed by breaking the loop into two different chains. Suppose we forget that the joint between $\mathcal{A}_5$ and $\mathcal{A}_6$ exists, as shown in Figure 3.27. Consider two different kinematic chains that start at the joint on the extreme left. There is an upper chain from $\mathcal{A}_1$ to $\mathcal{A}_5$ and a lower chain from $\mathcal{A}_{10}$ to $\mathcal{A}_6$. The transformations for any of these bodies can be obtained directly from the techniques of Section 3.3.1. Thus, it is easy to transform the bodies, but how do we choose parameter values that ensure $\mathcal{A}_5$ and $\mathcal{A}_6$ are connected at their common joint? Using the upper chain, the position of this joint may be expressed as

$$T_1(\theta_1)T_2(\theta_2)T_3(\theta_3)T_4(\theta_4)T_5(\theta_5) \begin{pmatrix} a_5 \\ 0 \\ 1 \end{pmatrix}, \qquad (3.79)$$

in which $(a_5, 0) \in \mathcal{A}_5$ is the location of the joint of $\mathcal{A}_5$ that is supposed to connect to $\mathcal{A}_6$. The position of this joint may also be expressed using the lower chain as

$$T_{10}(\theta_{10})T_9(\theta_9)T_8(\theta_8)T_7(\theta_7)T_6(\theta_6) \begin{pmatrix} a_6 \\ 0 \\ 1 \end{pmatrix}, \qquad (3.80)$$

with $(a_6, 0)$ representing the position of the joint in the body frame of $\mathcal{A}_6$. If the loop does not have to be maintained, then any values for $\theta_1, \ldots, \theta_{10}$ may be selected, resulting in ten degrees of freedom. However, if a loop must maintained,

then (3.79) and (3.80) must be equal,

$$
T_1(\theta_1)T_2(\theta_2)T_3(\theta_3)T_4(\theta_4)T_5(\theta_5)\begin{pmatrix} a_5 \\ 0 \\ 1 \end{pmatrix} = T_{10}(\theta_{10})T_9(\theta_9)T_8(\theta_8)T_7(\theta_7)T_6(\theta_6)\begin{pmatrix} a_6 \\ 0 \\ 1 \end{pmatrix},
$$

(3.81)

which is quite a mess of nonlinear, trigonometric equations that must be solved. The set of solutions to (3.81) could be very complicated. For the example, the true degrees of freedom is eight because two were removed by making the joint common. Since the common joint allows the links to rotate, exactly two degrees of freedom are lost. If $\mathcal{A}_5$ and $\mathcal{A}_6$ had to be rigidly attached, then the total degrees of freedom would be only seven. For most problems that involve loops, it will not be possible to obtain a nice parameterization of the set of solutions. This a form of the well-known *inverse kinematics problem* [4, 18, 23, 28].

In general, a complicated arrangement of links can be imagined in which there are many loops. Each time a joint along a loop is "ignored," as in the procedure just described, then one less loop exists. This process can be repeated iteratively until there are no more loops in the graph. The resulting arrangement of links will be a tree for which the previous techniques of this section may be applied. However, for each joint that was "ignored" an equation similar to (3.81) must be introduced. All of these equations must be satisfied simultaneously to respect the original loop constraints. Suppose that a set of value parameters is already given. This could happen, for example, using motion capture technology to measure the position and orientation of every part of a human body in contact with the ground. From this the solution parameters could be computed, and all of the transformations are easy to represent. However, as soon as the model moves, it is difficult to ensure that the new transformations respect the closure constraints. The foot of the digital actor may push through the floor, for example. Further information on this problem appears in Section 4.4.

## 3.5 Nonrigid Transformations

One can easily imagine motion planning for nonrigid bodies. This falls outside of the families of transformations studied so far in this chapter. Several kinds of nonrigid transformations are briefly surveyed here.

**Linear transformations** A rotation is a special case of a linear transformation, which is generally expressed by an $n \times n$ matrix, $M$, assuming the transformations are performed over $\mathbb{R}^n$. Consider transforming a point $(x, y)$ in a 2D robot, $\mathcal{A}$, as

$$
\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}.
$$

(3.82)

If $M$ is a rotation matrix, then the size and shape of $\mathcal{A}$ will remain the same. In some applications, however, it may be desirable to distort these. The robot can
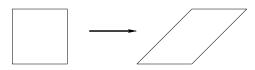
Figure 3.28: Shearing transformations may be performed.

be *scaled* by $m_{11}$ along the $x$-axis and $m_{22}$ along the $y$-axis by applying

$$
\begin{pmatrix} m_{11} & 0 \\ 0 & m_{22} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix},
$$

(3.83)

for positive real values $m_{11}$ and $m_{22}$. If one of them is negated, then a mirror image of $\mathcal{A}$ is obtained. In addition to scaling, $\mathcal{A}$ can be *sheared* by applying

$$
\begin{pmatrix} 1 & m_{12} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}
$$

(3.84)

for $m_{12} \neq 0$. The case of $m_{12} = 1$ is shown in Figure 3.28.

The scaling, shearing, and rotation matrices may be multiplied together to yield a general transformation matrix that explicitly parameterizes each effect. It is also possible to extend the $M$ from $n \times n$ to $(n + 1) \times (n + 1)$ to obtain a homogeneous transformation matrix that includes translation. Also, the concepts extend in a straightforward way to $\mathbb{R}^3$ and beyond. This enables the additional effects of scaling and shearing to be incorporated directly into the concepts from Sections 3.2-3.4.

**Flexible materials** In some applications there is motivation to move beyond linear transformations. Imagine trying to warp a flexible material, such as a mattress, through a doorway. The mattress could be approximated by a 2D array of links; however, the complexity and degrees of freedom would be too cumbersome. For another example, suppose that a snake-like robot is designed by connecting 100 revolute joints together in a chain. The tools from Section 3.3 may be used to transform it with 100 rotation parameters, $\theta_1$, ..., $\theta_{100}$, but this may become unwieldy for use in a planning algorithm. An alternative is to approximate the snake with a deformable curve or shape.

For problems such as these, it is desirable to use a parameterized family of curves or surfaces. Spline models are often most appropriate because they are designed to provide easy control over the shape of a curve through the adjustment of a small number of parameters. Other possibilities include the generalized-cylinder and superquadric models that were mentioned in Section 3.1.3.

One complication is that complicated constraints may be imposed on the space of allowable parameters. For example, each joint of a snake-like robot could have a

small range of rotation. This would be easy to model using a kinematic chain; however, determining which splines from a spline family satisfy this extra constraint may be difficult. Likewise, for manipulating flexible materials, there are usually complicated constraints based on the elasticity of the material. Even determining its correct shape under the application of some forces requires integration of an elastic energy function over the material [16].

## Further Reading

Section 3.1 barely scratches the surface of geometric modeling. Most literature focuses on parametric curves and surfaces [8, 21, 24]. These models are not as popular for motion planning because obtaining efficient collision detection is most important in practice, and processing implicit algebraic surfaces is most important in theoretical methods. A thorough coverage of solid and boundary representations, including semi-algebraic models, can be found in [11]. Theoretical algorithm issues regarding semi-algebraic models are covered in [19, 20]. For a comparison of the doubly connected edge list to its variants, see [12].

The material of Section 3.2 appears in virtually any book on robotics, computer vision, or computer graphics. Consulting linear algebra texts may be helpful to gain more insight into rotations. There are many ways to parameterize the set of all 3D rotation matrices. The yaw-pitch-roll formulation was selected because it is the easiest to understand. There are generally 12 different variants of the yaw-pitch-roll formulation (also called *Euler angles*) based on different rotation orderings and axis selections. This formulation, however, is not well suited for the development of motion planning algorithms. It is easy (and safe) to use for making quick 3D animations of motion planning output, but it incorrectly captures the structure of the state space for planning algorithms. Section 4.2 introduces the quaternion parameterization, which correctly captures this state space; however, it is harder to interpret when constructing examples. Therefore, it is helpful to understand both. In addition to Euler angles and quaternions, there is still motivation for using many other parameterizations of rotations, such as spherical coordinates, Cayley-Rodrigues parameters, and stereographic projection. Chapter 5 of [3] provides extensive coverage of 3D rotations and different parameterizations.

The coverage in Section 3.3 of transformations of chains of bodies was heavily influenced by two classic robotics texts [4, 23]. The DH parameters were introduced in [9] and later extended to trees and loops in [13]. An alternative to DH parameters is exponential coordinates [22], which simplify some computations; however, determining the parameters in the modeling stage may be less intuitive. A fascinating history of mechanisms appears in [10]. Other texts on kinematics include [1, 6, 14, 17]. The standard approach in many robotics books [7, 26, 27, 28] is to introduce the kinematic chain formulations and DH parameters in the first couple of chapters, and then move on to topics that are crucial for controlling robot manipulators, including dynamics modeling, singularities, manipulability, and control. Since this book is concerned instead with planning algorithms, we depart at the point where dynamics would usually be covered and move into a careful study of the configuration space in Chapter 4.

## Exercises

1. Define a semi-algebraic model that removes a triangular "nose" from the region shown in Figure 3.4.

2. For distinct values of yaw, pitch, and roll, it is possible to generate the same rotation. In other words, $R(\alpha, \beta, \gamma) = R(\alpha', \beta', \gamma')$ for some cases in which at least $\alpha \neq \alpha$, $\beta \neq \beta'$, or $\gamma \neq \gamma'$. Characterize the sets of angles for which this occurs.

3. Using rotation matrices, prove that 2D rotation is commutative but 3D rotation is not.

4. An alternative to the yaw-pitch-roll formulation from Section 3.2.3 is considered here. Consider the following Euler angle representation of rotation (there are many other variants). The first rotation is $R_z(\gamma)$, which is just (3.39) with $\alpha$ replaced by $\gamma$. The next two rotations are identical to the yaw-pitch-roll formulation: $R_y(\beta)$ is applied, followed by $R_z(\alpha)$. This yields $R_{euler}(\alpha, \beta, \gamma) = R_z(\alpha) R_y(\beta) R_z(\gamma)$.

   (a) Determine the matrix $R_{euler}$.

   (b) Show that $R_{euler}(\alpha, \beta, \gamma) = R_{euler}(\alpha - \pi, -\beta, \gamma - \pi)$.

   (c) Suppose that a rotation matrix is given as shown in (3.43). Show that the Euler angles are

   $$\alpha = \text{atan2}(r_{23}, r_{13}), \qquad (3.85)$$

   $$\beta = \text{atan2}(\sqrt{1 - r_{33}^2}, r_{33}), \qquad (3.86)$$

   and

   $$\gamma = \text{atan2}(r_{32}, -r_{31}). \qquad (3.87)$$

5. There are 12 different variants of yaw-pitch-roll (or Euler angles), depending on which axes are used and the order of these axes. Determine all of the possibilities, using only notation such as $R_z(\alpha) R_y(\beta) R_z(\gamma)$ for each one. Give brief arguments that support why or why not specific combinations of rotations are included in your list of 12.

6. Let $\mathcal{A}$ be a unit disc, centered at the origin, and $\mathcal{W} = \mathbb{R}^2$. Assume that $\mathcal{A}$ is represented by a single, algebraic primitive, $H = \{(x, y) \mid x^2 + y^2 \leq 1\}$. Show that the transformed primitive is unchanged after any rotation is applied.

7. Consider the articulated chain of bodies shown in Figure 3.29. There are three identical rectangular bars in the plane, called $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$. Each bar has width 2 and length 12. The distance between the two points of attachment is 10. The first bar, $\mathcal{A}_1$, is attached to the origin. The second bar, $\mathcal{A}_2$, is attached to $\mathcal{A}_1$, and $\mathcal{A}_3$ is attached to $\mathcal{A}_2$. Each bar is allowed to rotate about its point of attachment. The configuration of the chain can be expressed with three angles,
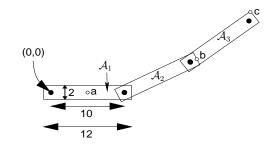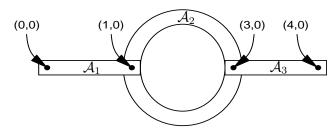
Figure 3.29: A chain of three bodies.



Figure 3.30: Another exercise involving a chain of bodies.

$(\theta_1, \theta_2, \theta_3)$. The first angle, $\theta_1$, represents the angle between the segment drawn between the two points of attachment of $\mathcal{A}_1$ and the $x$-axis. The second angle, $\theta_2$, represents the angle between $\mathcal{A}_2$ and $\mathcal{A}_1$ ($\theta_2 = 0$ when they are parallel). The third angle, $\theta_3$, represents the angle between $\mathcal{A}_3$ and $\mathcal{A}_2$. Suppose the configuration is $(\pi/4, \pi/2, -\pi/4)$.

(a) Use the homogeneous transformation matrices to determine the locations of points $a$, $b$, and $c$.

(b) Characterize the set of all configurations for which the final point of attachment (near the end of $\mathcal{A}_3$) is at $(0,0)$ (you should be able to figure this out without using the matrices).

8. A three-link chain of bodies that moves in a 2D world is shown Figure 3.30. The first link, $\mathcal{A}_1$, is attached at $(0,0)$ but can rotate. Each remaining link is attached to another link with a revolute joint. The second link, $\mathcal{A}_2$, is a rigid ring, and the other two links are rectangular bars.

Assume that the structure is shown in the zero configuration. Suppose that the linkage is moved to the configuration $(\theta_1, \theta_2, \theta_3) = (\frac{\pi}{4}, \frac{\pi}{2}, \frac{\pi}{4})$, in which $\theta_1$ is the angle of $\mathcal{A}_1$, $\theta_2$ is the angle of $\mathcal{A}_2$ with respect to $\mathcal{A}_1$, and $\theta_3$ is the angle of $\mathcal{A}_3$ with respect to $\mathcal{A}_2$. Using homogeneous transformation matrices, compute the position of the point at $(4,0)$ in Figure 3.30, when the linkage is at configuration $(\frac{\pi}{4}, \frac{\pi}{2}, \frac{\pi}{4})$ (the point is attached to $\mathcal{A}_3$).

9. Approximate a spherical joint as a chain of three short, perpendicular links that are attached by revolute joints and give the sequence of transformation matrices. Show that as the link lengths approach zero, the resulting sequence of transformation matrices converges to exactly representing the freedom of a spherical joint. Compare this approach to directly using a full rotation matrix, (3.42), to represent the joint in the homogeneous transformation matrix.

10. Figure 3.12 showed six different ways in which 2D surfaces can slide with respect to each other to produce a joint.

(a) Suppose that two bodies contact each other along a one-dimensional curve. Characterize as many different kinds of "joints" as possible, and indicate the degrees of freedom of each.

(b) Suppose that the two bodies contact each other at a point. Indicate the types of rolling and sliding that are possible, and their corresponding degrees of freedom.

11. Suppose that two bodies form a screw joint in which the axis of the central axis of the screw aligns with the $x$-axis of the first body. Determine an appropriate homogeneous transformation matrix to use in place of the DH matrix. Define the matrix with the screw radius, $r$, and displacement-per-revolution, $d$, as parameters.

12. Recall Example 3.6. How should the transformations be modified so that the links are in the positions shown in Figure 3.25 at the zero configuration ($\theta_i = 0$ for every revolute joint whose angle can be independently chosen)?

13. Generalize the shearing transformation of (3.84) to enable shearing of 3D models.

**Implementations**

14. Develop and implement a kinematic model for 2D linkages. Enable the user to display the arrangement of links in the plane.

15. Implement the kinematics of molecules that do not have loops and show them graphically as a "ball and stick" model. The user should be able to input the atomic radii, bond connections, bond lengths, and rotation ranges for each bond.

16. Design and implement a software system in which the user can interactively attach various links to make linkages that resemble those possible from using Tinkertoys (or another popular construction set that allows pieces to move). There are several rods of various lengths, which fit into holes in the center and around the edge of several coin-shaped pieces. Assume that all joints are revolute. The user should be allowed to change parameters and see the resulting positions of all of the links.

17. Construct a model of the human body as a tree of links in a 3D world. For simplicity, the geometric model may be limited to spheres and cylinders. Design and implement a system that displays the virtual human and allows the user to click on joints of the body to enable them to rotate.

18. Develop a simulator with 3D graphics for the Puma 560 model shown in Figure 3.4.

# Bibliography

[1] J. Angeles. *Spatial Kinematic Chains. Analysis, Synthesis, and Optimisation.* Springer-Verlag, Berlin, 1982.

[2] B. Armstrong, O. Khatib, and J. Burdick. The explicit dynamic model and inertial parameters of the Puma 560 arm. In *Proceedings IEEE International Conference on Systems, Man, & Cyberetics*, pages 510–518, 1986.

[3] G. S. Chirikjian and A. B. Kyatkin. *Engineering Applications of Noncommutative Harmonic Analysis.* CRC Press, Boca Raton, FL, 2001.

[4] J. J. Craig. *Introduction to Robotics.* Addison-Wesley, Reading, MA, 1989.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications, 2nd Ed.* Springer-Verlag, Berlin, 2000.

[6] A. G. Erdman, G. N. Sandor, and S. Kota. *Mechanism Design: Analysis and Synthesis, 4th Ed., Vol. 1.* Prentice Hall, Englewood Cliffs, NJ, 2001.

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

[8] J. Gallier. *Curves and Surfaces in Geometric Modeling.* Morgan Kaufmann, San Francisco, CA, 2000.

[9] R. S. Hartenberg and J. Denavit. A kinematic notation for lower pair mechanisms based on matrices. *Journal of Applied Mechanics*, 77:215–221, 1955.

[10] R. S. Hartenberg and J. Denavit. *Kinematic Synthesis of Linkages.* McGraw-Hill, New York, 1964.

[11] C. M. Hoffmann. *Geometric and Solid Modeling.* Morgan Kaufmann, San Francisco, CA, 1989.

[12] L. Kettner. Designing a data structure for polyhedral surfaces. In *Proceedings ACM Symposium on Computational Geometry*, pages 146–154, 1998.

[13] W. Khalil and J. F. Kleinfinger. A new geometric notation for open and closed-loop robots. In *Proceedings IEEE International Conference on Robotics & Automation*, volume 3, pages 1174–1179, 1986.

[14] J. T. Kimbrell. *Kinematic Analysis and Synthesis.* McGraw-Hill, New York, 1991.

[15] K. Kozlowski, P. Dutkiewicz, and W. Wróblewski. *Modeling and Control of Robots.* Wydawnictwo Naukowe PWN, Warsaw, Poland, 2003. In Polish.

[16] F. Lamiraux and L. Kavraki. Path planning for elastic plates under manipulation constraints. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 151–156, 1999.

[17] J. M. McCarthy. *Geometric Design of Linkages.* Springer-Verlag, Berlin, 2000.

[18] J.-P. Merlet. *Parallel Robots.* Kluwer, Boston, MA, 2000.

[19] B. Mishra. *Algorithmic Algebra.* Springer-Verlag, New York, 1993.

[20] B. Mishra. Computational real algebraic geometry. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 537–556. CRC Press, New York, 1997.

[21] M. E. Mortenson. *Geometric Modeling, 2nd Ed.* Wiley, New York, 1997.

[22] R. M. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, Boca Raton, FL, 1994.

[23] R. P. Paul. *Robot Manipulators: Mathematics, Programming, and Control.* MIT Press, Cambridge, MA, 1981.

[24] L. Piegl. On NURBS: A survey. *IEEE Transactions on Computer Graphics & Applications*, 11(1):55–71, Jan 1991.

[25] H. Pottman and J. Wallner. *Computational Line Geometry.* Springer-Verlag, Berlin, 2001.

[26] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators.* Springer-Verlag, Berlin, 1996.

[27] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control.* Wiley, New York, 2005.

[28] T. Yoshikawa. *Foundations of Robotics: Analysis and Control.* MIT Press, Cambridge, MA, 1990.