Chapter 3

Kinematics

Kinematics is a geometric approach to robot motion. It is the study of positions and angles and their rates of change. In kinematics, we study abstractions that simplify our analysis of the motion of billions of particles into a small number of motions that we have designed into the system. The primary abstractions are *joints*, which permit motion, and *links*, which are *rigid bodies*, thus unbendable. In a classic mechanical system such as this, we call the joint angles *freedoms*, and we count the number of *degrees of freedom* of the whole system such that a vector can completely describe the configuration of the robot (and the positions and motions of the billions of particles) at some moment in time.

We start with the basics of rigid body motion, namely translation and rotation. Then we introduce the configuration space and define forward and inverse kinematic mappings, which address questions of where the robot is and where we want it to be. Lastly, we turn to velocity kinematics, which relates the rates of different parameterizations of a robot's position.

It should be noted that kinematics is not concerned with *why* things move; that question is addressed in mechanics by the introduction of forces, which induce accelerations. In kinematics, we are only concerned with position and its first derivative, velocity. When we talk about velocity kinematics, we may be describing the rate of change of position of an object with respect to time or with respect to that motion expressed by other coordinates.

3.1 The Rigid Body Assumption

A *rigid body* is a set of particles that move together because they are attached. The distance between any two particles remains fixed, regardless of any motions of the body or forces exerted upon it. When one particle in a rigid body is pushed, the other particles will all follow in some coherent way. In other words, a rigid body is an abstraction: an idealized, completely undeformable solid body. We use the *rigid body assumption* in robotics to model a robot's whole body or its components, such as the links of a manipulator. Since no object is completely rigid, the rigid body assumption holds only under a set of other reasonable assumptions: no unusually-large forces or deformations are imparted on the object; the object is not broken, cut, or otherwise disassembled.

The power of the rigid body abstraction is in its ease of description. Since the particles do not move with respect to one another, **knowing a single position and orientation fully specifies the position of every particle in the rigid body.**

Examples of objects that fit the rigid body assumption well include a brick, a metal crowbar, and a lightbulb. Some objects that fit the rigid body assumption poorly include a piece of paper (it bends too easily), a body of water (the particles freely rearrange and flow), and a Jenga tower (the blocks fall apart easily). Some objects are in between these two extremes, and we may choose to apply the rigid body assumption anyway, knowing that it is imperfect. For example, a beach ball stays mostly spherical, but it is soft and deforms where it touches objects. A building is rigid and largely fixed, but the doors can swing independently of the building; we just neglect this detail due to the difference in scale.

When we talk about a robot, it is important to be clear what assumptions we are making for the sake of a clean mathematical analysis. In order to describe a rigid body mathematically, we need to introduce some formalisms.

A *reference frame* is a set of axes fixed to a point, or *particle*. The axes of the reference frame form a linear basis by which to describe the positions of points. The reference frame shown in Fig. 3.1, $\{G\}$, represents a *global reference frame*, i.e. a coordinate frame fixed in a 3-dimensional Cartesian space $\mathbb{R}^3$ at a point O_G with axes defined by the unit vectors $\{\hat{x}_G, \hat{y}_G, \hat{z}_G\}$. O_G is called the *origin* of frame G. By convention, subscripts are names, such as the B-origin or the G-origin.

The kinematic state of a particle A can be fully specified by its position with respect to $\{G\}$. We can write the position of A with respect to $\{G\}$ as a 3-vector,

$$P_A^G = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T, \tag{3.1}$$

which is called a *position vector*. Vectors are always columns, but we have written the vector in transposed form for compactness. The components of the position vector denote the projection of P_A^G onto each axis of $\{G\}$ (see Fig. 3.1). In 2D, the position vector would have two components, as $P_A^G = \begin{bmatrix} p_x & p_y \end{bmatrix}^T$. The superscript and subscript of P have special meanings. The superscript denotes the reference frame in which the position of this point is expressed, and the subscript provides the name of the point whose position is being referred to.

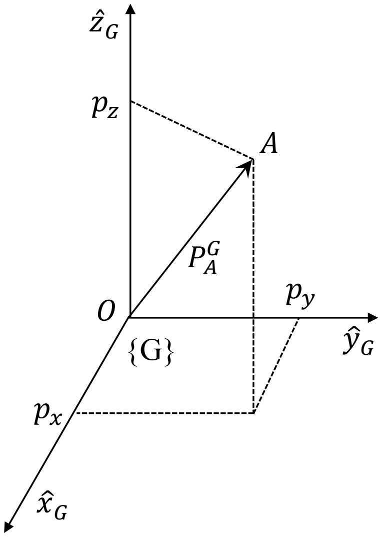


Figure 3.1: A position vector P_A^G, defined with respect to the reference frame $\{G\}$.

In Fig. 3.2, we see that the position of frame $\{B\}$ is expressed with respect to the global frame $\{G\}$ as

$$P_{O_B}^G = \begin{bmatrix} q_x^G & q_y^G & q_z^G \end{bmatrix}^T . \tag{3.2}$$

We will often write P_B^G as a shorthand for $P_{O_B}^G$, but it is important to remember that the position of a frame is the position of its origin. Here, the coordinates (q_x^G, q_y^G, q_z^G) describe the position of frame $\{B\}$ in $\{G\}$-frame coordinates. Since B is a rigid body that can move with respect to the global frame, these coordinates can vary. To find the values of the coordinates, we can project the point O_B onto the $\{G\}$-frame axes $\hat{x}_G$, $\hat{y}_G$, and $\hat{z}_G$. Our assumption that B is a rigid body means that a motion of O_B causes a motion of the whole frame. Furthermore, we can pick any arbitrary point A on the rigid body B and express its coordinates in the frame $\{B\}$ as a constant position vector

$$P_A^B = \begin{bmatrix} p_x^B & p_y^B & p_z^B \end{bmatrix}^T . \tag{3.3}$$

This expression says that the position of the point A in the reference frame $\{B\}$ is defined by the coordinates (p_x^B, p_y^B, p_z^B), as a linear combination of the axes of the B frame. We can always express a point in a different basis by using a different reference frame; for example, the position vectors P_A^B and P_A^G both refer to the position of the same point named A but are defined with respect to different

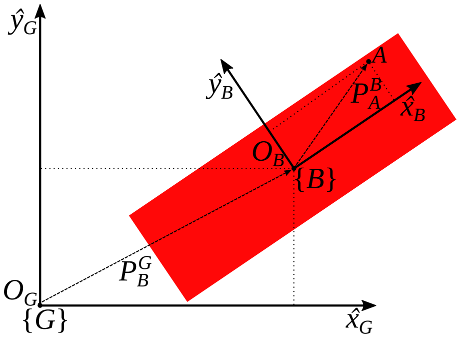


Figure 3.2: The body-fixed reference frame $\{B\}$ and the global-fixed reference frame $\{G\}$. A point A is defined with respect to the body frame. Since it is a rigid body, the position of A in the $\{B\}$ frame, expressed as P_A^B, is fixed. However, the position of the $\{B\}$ frame in the $\{G\}$ frame, expressed as P_B^G, can change if the rigid body moves. Note that such motion does not change the value of P_A^B.

frames. As a result, unless $\{B\}$ and $\{G\}$ are coincident, P_A^B and P_A^G have different values. Note that $P_A^B \neq P_A^G$ implies $\{B\} \neq \{G\}$, but the converse is not true.

The motion of a rigid body in space can be fully described by its position with respect to a coordinate frame at every instant in time. A *rigid motion* of a body is a continuous movement of its particles under the constraint that that the distance between any two particles remains invariant. The instantaneous net movement of a body via rigid motion is called a *rigid displacement*, which may result from *translation* (linear displacement, i.e. a change in position), *rotation* (angular displacement, i.e. a change in orientation), or both at the same time.

For example, if the rigid body B starts at the global origin O_G and then moves to a new position, we say that it was translated to the new position by a motion of P_B^G. The terms position and translation are often used interchangeably when we are only concerned about the current position of B rather than the history of motions that were applied to B; its current position is the sum of all the translation operations that were applied to it. Similarly, the terms rotation and orientation are sometimes used synonymously because you can combine rotations to define a rigid body's current orientation. The key concept here is that **any arbitrary sequence of translation and rotation operations can be combined and expressed as a single translation and a single rotation**. Furthermore, if we do not restrict to rotating about the origin of the reference frame, then any arbitrary combined rotation and translation can be expressed as a *pure rotation* about some point. Fig. 3.3 illustrates these concepts.

Note that a particle can only translate; it cannot rotate because it has no orientation. However, when we define a rigid body's translation and rotation, we define

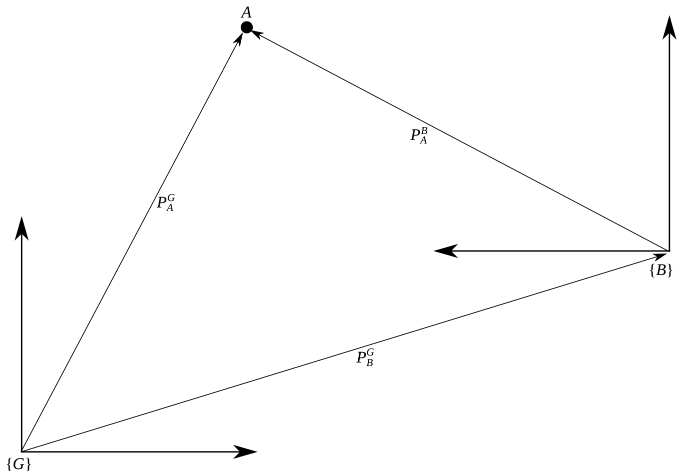


Figure 3.3: The body frame $\{B\}$ of the robot is displaced from the global frame $\{G\}$ by both a translation of $(3, 1)$ meters and a rotation of $\frac{\pi}{2}$ radians about the origin of $\{B\}$. Note that $P_A^B = P_A^G$. In other words, when the robot moved from $\{G\}$ to $\{B\}$, the point A appeared to stay fixed from its perspective at $P_A^G = P_A^B = (1, 2)$. Every rigid body displacement has exactly one such fixed point (possibly located at infinity). Therefore, we can re-express the same rigid body displacement as a pure rotation about the point A by $\frac{\pi}{2}$ radians.

the position of every particle within the rigid body.

A translation has only one common representation, which is a vector that looks just like a position vector. The vector P_B^G can be interpreted as either (1) a position vector giving the location of the origin of the $\{B\}$ frame with respect to the $\{G\}$ frame, or (2) a translation of a rigid body into the $\{B\}$ frame from the $\{G\}$ frame. Translations are composed by vector addition.

Unlike translations, rotations are complex, and many different ways to represent them have been devised. Later on, we describe several of these representations as well as their advantages and disadvantages. First, we discuss a common type of robot for which the rigid body assumption is a good model: mobile robots.

3.2 Mobile Robot Kinematics

Many mobile robots can be modeled well as rigid bodies, like automated guided vehicles (AGVs), drones (UAVs or unmanned aerial vehicles), and automated underwater vehicles (AUVs). We focus here on the kinematics of ground vehicles, which use wheels to locomote and steer.

Since we assume mobile robots are rigid bodies, we can define the configuration of the robot by its coordinate frame. In 2D, the position of the robot is given by the position of the origin of its frame as $P_B^G = \begin{bmatrix} x & y \end{bmatrix}^T$. We need to measure the orientation as well. Although we have not addressed general rotations in 3D yet, we can represent a 2D orientation as an angle θ in radians measured between the x-axis of the global frame and the x-axis of the robot's body frame. By convention, a mobile robot's body-frame x-axis points forwards in the normal direction of travel. We may express a wheeled mobile robot's configuration as a vector $q = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$.

3.2.1 Mobile Robot Constraints

Constraints are key to a proper understanding of the motion of wheeled mobile robots. A kinematic model of a wheeled mobile robot allows us to determine how the robot moves based on the geometry of constraints imposed by its wheels. To find tractable equations for the model, we make several assumptions:

1. The robot is a single rigid body.
2. The robot moves on a smooth, flat surface.
3. No translational *slip* occurs between the robot's wheels and the floor; that is, the torque applied to the wheels does not exceed the available traction.
4. Wheels only spin about their central axis and – if they are steering wheels – about the $\hat{z}$-axis, which is perpendicular to the floor.

The three dimensions of the configuration vector q are position variables expressing the three degrees of freedom of a rigid body. Having three degrees of freedom, however, does not mean that a rigid body can always move instantaneously in all three directions. Nor does it mean that a robot can even reach points in all three directions eventually given infinite time.

Position constraints reduce reachable degrees of freedom, whereas *instantaneous degrees of freedom* are reduced by *velocity constraints*. That is, velocity constraints describe directions of unallowed motion. We will see that some velocity constraints are also position constraints, but others are not. For now, we will analyze only instantaneous motions. The motions (or lack thereof) expressed by instantaneous freedoms and velocity constraints are described as velocities that are non-zero or zero, respectively. A mobile robot's velocity is expressed as the vector

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}. \tag{3.4}$$

We first introduce the equations for modeling the velocity constraints and instantaneous freedoms of arbitrary wheeled mobile robots, and then we will consider how they apply to several robot designs.

Let the number of degrees of freedom be d and the number of instantaneous freedoms be $m \leq d$. We express the instantaneous freedoms as a set of m equations with $1 \leq i \leq m$,

$$\dot{q} = \sum_i g_i(q) u_i, \tag{3.5}$$

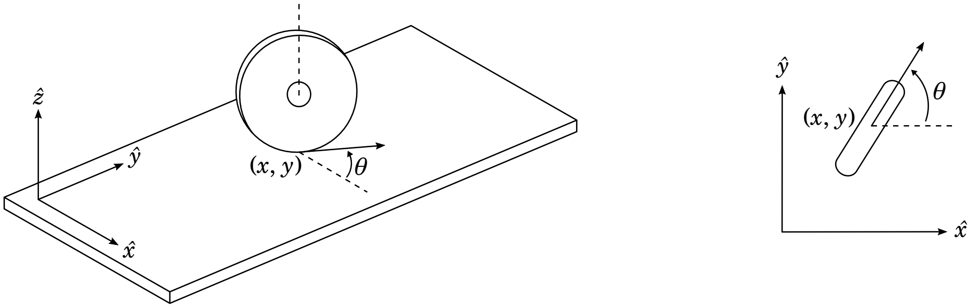
where the vectors $g_i(q) \in \mathbb{R}^3$ define the possible motions, and the u_i terms are the scalar control inputs. Thus, for each i, when the control u_i is applied to the mobile robot, it moves with velocity $g_i(q)u_i$. Furthermore, any linear combination of control inputs is possible.

Let the number of velocity constraints be $n \leq d$. Since constraints forbid motion, they are written as

$$\forall i, w_i(q) \cdot \dot{q} = 0. \tag{3.6}$$

$w_i(q)$ is a velocity vector in which motion is prohibited. Examples of both $g_i(q)$ and $w_i(q)$ are given below.

Unicycle Model



(a)

(b)

Figure 3.4: A unicycle rolling on a plane, shown from the side (a) and overhead (b).

The most basic mobile robot is a unicycle (see Fig. 3.4), which illustrates the basic instantaneous freedoms and velocity constraints of the wheel. The point of contact between the unicycle and the ground is (x, y), and its heading (positive x-axis) direction is θ. These are expressed in some global coordinate frame.

There are two control inputs to the unicycle:

1. u_1 is the forward-backward driving speed, and
2. u_2 is the heading direction turning speed.

A control input is a degree of freedom in which the robot is not only able to move instantaneously but can do so under its own power. The instantaneous freedoms associated with these controls are

$$g_1(q) = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} \qquad \text{and} \qquad g_2(q) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{3.7}$$

Note that the robot's x-axis is the unit vector $(1, 0)$ in its own body frame. That vector expressed in the global frame is $(\cos\theta, \sin\theta)$ for any orientation θ. The combined instantaneous freedoms achievable from these two control inputs are written as

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u_1\cos\theta \\ u_1\sin\theta \\ u_2 \end{bmatrix}. \tag{3.8}$$

Any values of the control inputs u_1 and u_2 produce a velocity that is kinematically allowed because it obeys the unicycle's (single) motion constraint. That constraint is

$$w_1(q) = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix}, \tag{3.9}$$

meaning that the unicycle cannot move sideways. Note that since the unicycle is prohibited to move instantaneously in the direction along its body-frame y-axis, we can derive the above constrained by rotating the mobile robot's body y-axis unit vector $(0, 1)$ into the global frame. Written out, this constraint is

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0. \tag{3.10}$$

Unsteered Cart

The cart shown in Fig. 3.5 has four wheels, all of which are fixed to the body of the cart. It therefore has no ability to steer.

Since the cart cannot turn, its only control input is the forward-backward driving speed. Hence, its instantaneous freedom can be expressed with respect to this lone control input as

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} u_1 = \begin{bmatrix} u_1\cos\theta \\ u_1\sin\theta \\ 0 \end{bmatrix}, \tag{3.11}$$

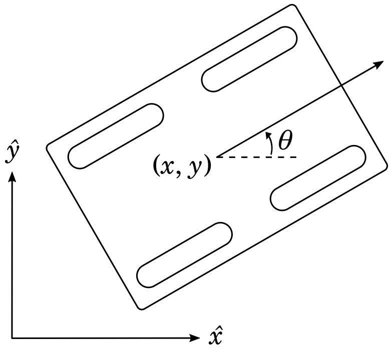


Figure 3.5: An unsteered cart cannot change its heading direction.

and its two velocity constraints are

$$w_1(q) = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} \qquad \text{and} \qquad w_2(q) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{3.12}$$

Thus, the constraint expressions are

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0 \tag{3.13}$$

$$\dot{\theta} = 0, \tag{3.14}$$

Nonholonomic Constraints

Note that the instantaneous freedoms of a mobile robot are collective, meaning that any linear combination of the u_i terms is an allowable motion, whereas the constraints are separate. A constraint is called *holonomic* (or integrable) if it depends only on the configuration q and not its derivative $\dot{q}$; that is, a holonomic constraint must be expressible as a function

$$f(q) = 0. \tag{3.15}$$

A constraint that does not meet the above criteria is called *nonholonomic*. If all of the constraints of a system are holonomic, it is considered a *holonomic system*; any other system is a *nonholonomic system*. Let us revisit the two models above.

Unicycle Model Constraints

Consider the single constraint against sideways motion in Equation 3.10, which we repeat here for clarity:

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0.$$

Since this constraint cannot be integrated (it has no expression in terms of only q's variables x, y, and θ), it is nonholonomic, and therefore the unicycle is a nonholonomic system.

Another way to check whether a system is holonomic or nonholonomic is to compare the number of instantaneous degrees of freedom to the *reachable* degrees of freedom. If the number of instantaneous degrees of freedom is **less than** the reachable degrees of freedom, the system is **nonholonomic**. If the number of instantaneous degrees of freedom is **equal to** the reachable degrees of freedom, the system is **holonomic**.

In the case of the unicycle, it can reach anywhere in the plane with any orientation. Thus, it has two controls but three reachable degrees of freedom (its x-position, its y-position, and its heading direction) and thus is nonholonomic.

Unsteered Cart Constraints

Recall Equation 3.13–3.14:

$$\begin{aligned} \dot{y}\cos\theta - \dot{x}\sin\theta &= 0 \\ \dot{\theta} &= 0, \end{aligned}$$

Compared to the unicycle, there is one additional constraint. The pair of these constraints *can* be integrated to a derivative-free form:

$$(y - y_0)\cos\theta_0 - (x - x_0)\sin\theta_0 = 0 \tag{3.16}$$

$$\theta = \theta_0. \tag{3.17}$$

Since these constraints can be integrated, they are both holonomic, and thus the unsteered cart is a holonomic system. Each holonomic constraint reduces the system's reachable degrees of freedom by one, so although the cart has an x-position, a y-position, and a heading direction θ, it only has one effective degree of freedom. Intuitively, we can see that the cart must always remain on a single line in the plane because it cannot turn, therefore restricting its motion to a single reachable degree of freedom. Finally, note that the number of controls (one) is equal to the reachable degrees of freedom (one), which confirms that unsteered cart is a holonomic system.

Thus, we see that in a holonomic system, velocity constraints on the robot's motion are also constraints on its position, meaning that the reachable degrees of freedom are less than the full degrees of freedom of the rigid body. In contrast, the velocity constraints in a nolholonomic system do not constrain its position, so the full degrees of freedom are reachable.

3.2.2 Mobile Robot Steering Kinematics

We can analyze the motion of a wheel in the plane by observing the pattern of freedoms and constraints. A wheel can move forward or backward, but it resists side-to-side motion. If all the wheels on a robot are parallel, as is the case with both the unicycle and the unsteered cart, then all the constraints of the individual wheels are consistent. Note, however, that in the unsteered cart each wheel's constraint against sideways translation forbids rotation of the other wheels. If the wheels are not parallel, such as in a bicycle, then we know from experience that the result is a controlled turn. However, we desire a formal approach to the analysis of these multi-wheel configurations.

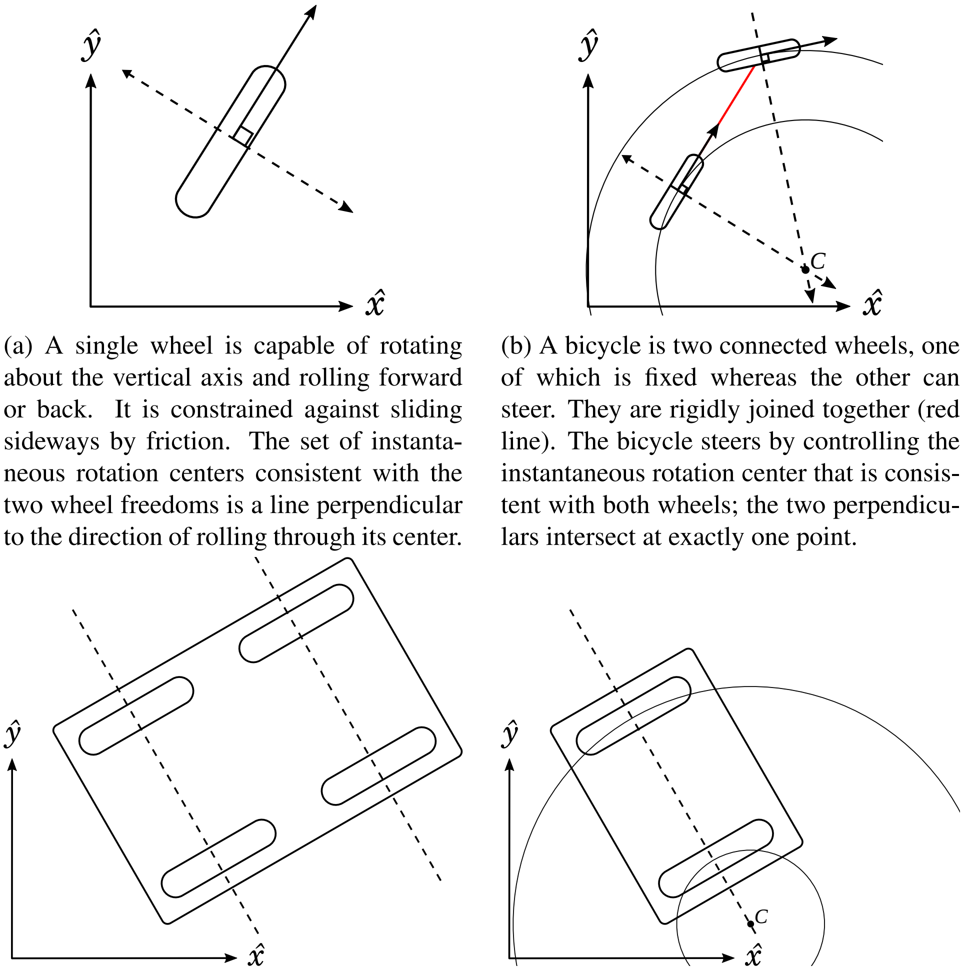
It is illuminating to look at the situation in terms of possible instantaneous rotation centers of a wheel, as in Fig. 3.6. Since a rigid body displacement can be described as a point to rotate about, then the instantaneous rigid body motion can be expressed as a pure rotation about some point[1], as shown in Fig. 3.3.

The instantaneous rotation center for a wheel depends on the values of the two u_i parameters in Equation 3.8. We will often refer to u_1 and u_2 by the names linear velocity (v) and angular velocity (ω), respectively. When the controls u_1 and u_2 are mixed, we get a combination of constant linear and angular velocities, causing the vehicle to follow a circular arc. If the controls are varied over time, then the vehicle can be made to follow arbitrary curves. However, in these notes, we focus our analysis on the instantaneous case, so we consider only constant linear and angular velocities. The ratio of linear and angular velocities control the curvature (κ) of the arc, according to the following relation:

$$\kappa = \frac{1}{r} = \frac{\omega}{v}. \tag{3.18}$$

Note that curvature of an arc is the reciprocal of the radius of curvature of that arc about the instantaneous center of rotation. The center of rotation is therefore defined by the interplay between linear and angular velocity. The set of arcs consistent with the nonholonomic constraint are those tangential to the wheel's x-axis at its origin. The locus of centers of such arcs is the set of points belonging to the line collinear with the y-axis. That is, the nonholonomic constraint of the wheel permits an instantaneous rotation center at any point along a line perpendicular to the direction of travel (see Fig. 3.6a) passing through the wheel's axle. A rotation center at the center of the wheel would correspond to turning in place (pure u_2). Any other point along the line would result in a curved path (some mix of u_1 and u_2). Rotation about the point at infinity (in either direction) corresponds to a straight motion of the wheel with no turning (pure u_1).

[1] Driving straight is a rotation about a point infinitely far away.



(a) A single wheel is capable of rotating about the vertical axis and rolling forward or back. It is constrained against sliding sideways by friction. The set of instantaneous rotation centers consistent with the two wheel freedoms is a line perpendicular to the direction of rolling through its center.

(b) A bicycle is two connected wheels, one of which is fixed whereas the other can steer. They are rigidly joined together (red line). The bicycle steers by controlling the instantaneous rotation center that is consistent with both wheels; the two perpendiculars intersect at exactly one point.

(c) The perpendiculars of the unsteered four-wheeled cart form parallel lines that meet at infinity. Thus, the four-wheeled cart can only move in a straight line (a translation can be regarded as a rotation about a point at infinity).

(d) The two-wheeled cart permits steering by turning the wheels at different rates. Thus, it is generally referred to as differential drive.

Figure 3.6: Instantaneous rotation centers of wheels can be any point along the perpendicular.

In a bicycle (Fig. 3.6b), the wheels' two perpendicular axes always meet at exactly one point that kinematically defines the steering direction of the bicycle. Thus, when the wheels are parallel, the bicycle travels straight. Otherwise, it curves in an arc.

The analysis by instantaneous rotation centers gives us new insight into why the unsteered cart in Fig. 3.5 can only move in a straight line, since the four lines of instantaneous rotation centers meet only at infinity (Fig. 3.6c). Note that a translation is equivalent to a rotation about a point at infinity. There is a class of mobile robots called *skid-steered vehicles* that are built with four fixed wheels, like the unsteered cart, for ruggedness. They steer by violating the nonholonomic constraint of two or more wheels by sliping sideways. Such vehicles generally traverse rough terrain like mud and rocks where there is not always good frictional contact with the ground.

Systems of Wheel Constraints

Having considered several layouts of mobile robot wheels, one might wonder how in general to identify whether a given layout would result in motion. Suppose we are given a set of n wheels fixed in the frame of the rigid body mobile robot. For each wheel $1 \leq i \leq n$, we have a coordinate frame's origin (p_{x_i}, p_{y_i}) as well as the directions of the unit vector x-axis (x_{x_i}, x_{y_i}) and y-axis (y_{x_i}, y_{y_i}).

We have noted that any two lines meet at a point (possibly at infinity). However, for greater than two lines, the intersections may not be coincident. To find out, we can set up a system of equations. A pair of equations constrains the x- and y-coordinates of one intersection point. Given n constraints, one per wheel, we define a parameter vector $t = \begin{bmatrix} t_1 & t_2 & \dots & t_n \end{bmatrix}^T$.

Now we set up and solve the system of linear equations $At = b$, where

$$A = \begin{bmatrix} y_{x_1} & -y_{x_2} & 0 & \dots & 0 \\ y_{y_1} & -y_{y_2} & 0 & \dots & 0 \\ y_{x_1} & 0 & -y_{x_3} & \dots & 0 \\ y_{y_1} & 0 & -y_{y_3} & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ y_{x_1} & 0 & 0 & \dots & -y_{x_n} \\ y_{y_1} & 0 & 0 & \dots & -y_{y_n} \end{bmatrix} \tag{3.19}$$

$$b = \begin{bmatrix} p_{x_1} - p_{x_2} \\ p_{y_1} - p_{y_2} \\ p_{x_1} - p_{x_3} \\ p_{y_1} - p_{y_3} \\ \vdots \\ p_{x_1} - p_{x_n} \\ p_{y_1} - p_{y_n} \end{bmatrix} \tag{3.20}$$

Thus, we have $2(n-1)$ constraints and n unknowns. For $n > 2$, the system appears overconstrained; however, coincident rotation centers are redundant constraints. Therefore, we can attempt to solve the system of equations and then check if we got an exact solution. Since A is not in general invertible, we must instead use the left pseudoinverse to find a least-squared-error solution to an overconstrained system. The left pseudoinverse is $A^+ = (A^T A)^{-1} A^T$. Therefore,

$$At = b \tag{3.21}$$

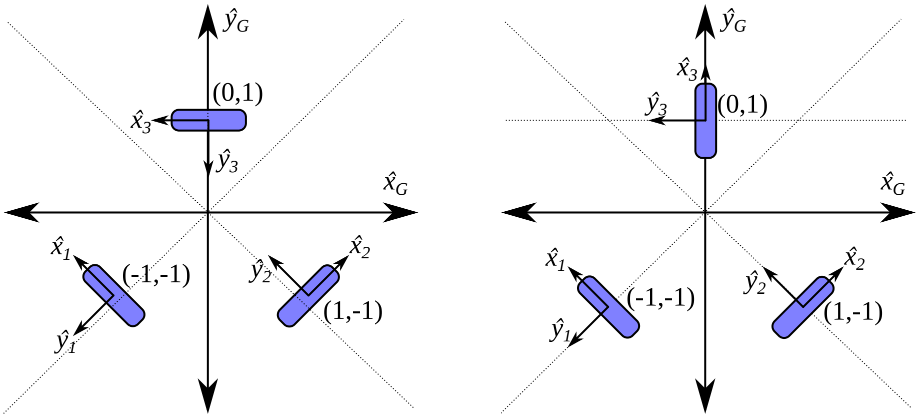
$$A^+ At = A^+ b \tag{3.22}$$

$$(A^T A)^{-1} A^T At = (A^T A)^{-1} A^T b \tag{3.23}$$

$$t = (A^T A)^{-1} A^T b \tag{3.24}$$

The solution value of $At = b$ can be obtained in Matlab or Octave using the left pseudoinverse with the notation $t = A \backslash b$. Having found an approximate solution t, we can now check if it perfectly solves the problem by computing $At - b$. An exact solution will be indicated by a vector of all zeros. Thus, a single instantaneous center of rotation exists that is consistent with all wheel constraints. If the resulting vector is non-zero, then the wheel constraints are entirely contradictory and this configuration of wheels is immobilized due to constraints.

We consider two examples of tricycles, shown in Fig. 3.7. In Fig. 3.7a, all three wheel constraints are consistent. To show that, consider the table of values:



(a) This tricycle can spin in place because all three lines coincident with the y-axes, representing the three wheel constraints, meet at the origin.

(b) This tricycle cannot move because there is no single point where all the y-axes meet. Each pair of wheels wants to rotate about a distinct point.

Figure 3.7: One can analyze sets of wheel velocity constraints by drawing the lines coincident with the y-axes of the wheels and solving for a point where all lines meet; this is the instantaneous center of rotation.

Wheel	Position	y-axis
1	$(-1,-1)$	$(-1,-1)$
2	$(1,-1)$	$(-1,1)$
3	$(0,1)$	$(0,-1)$

From the table, we can find

$$A = \begin{bmatrix} y_{x_1} & -y_{x_2} & 0 \\ y_{y_1} & -y_{y_2} & 0 \\ y_{x_1} & 0 & -y_{x_3} \\ y_{y_1} & 0 & -y_{y_3} \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ -1 & -1 & 0 \\ -1 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.25}$$

$$b = \begin{bmatrix} p_{x_1} - p_{x_2} \\ p_{y_1} - p_{y_2} \\ p_{x_1} - p_{x_3} \\ p_{y_1} - p_{y_3} \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \\ -1 \\ -2 \end{bmatrix} \tag{3.26}$$

Solving for t via the left pseudoinverse, we get,

$$At = b \tag{3.27}$$

$$t = (A^T A)^{-1} A^T b \tag{3.28}$$

$$= \begin{bmatrix} 1.3333 \\ -0.5 \\ -0.6667 \end{bmatrix}. \tag{3.29}$$

Now, to check that this answer works, we compute

$$At - b = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T.$$

Therefore, the wheel constraints are consistent and the robot can turn about the origin.

If we slightly modify this example as shown in Fig. 3.7b, we can see that the y-axis lines no longer meet at a single point, and therefore the wheel constraints contradict one another. Consider the table of values:

Wheel	Position	y-axis
1	$(-1,-1)$	$(-1,-1)$
2	$(1,-1)$	$(-1,1)$
3	$(0,1)$	**(-1,0)**

From the table, we can find

$$A = \begin{bmatrix} y_{x_1} & -y_{x_2} & 0 \\ y_{y_1} & -y_{y_2} & 0 \\ y_{x_1} & 0 & -y_{x_3} \\ y_{y_1} & 0 & -y_{y_3} \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ -1 & -1 & 0 \\ -1 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix} \tag{3.30}$$

$$b = \begin{bmatrix} p_{x_1} - p_{x_2} \\ p_{y_1} - p_{y_2} \\ p_{x_1} - p_{x_3} \\ p_{y_1} - p_{y_3} \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \\ -1 \\ -2 \end{bmatrix} \tag{3.31}$$

Once again solving for t via the left pseudoinverse, we get,

$$At = b \tag{3.32}$$

$$t = (A^T A)^{-1} A^T b \tag{3.33}$$

$$= \begin{bmatrix} 1.3333 \\ -1 \\ -0.3333 \end{bmatrix}. \tag{3.34}$$

To check that this answer works, we compute

$$At - b = \begin{bmatrix} -0.3333 & -0.3333 & 0 & 0.6667 \end{bmatrix}^T .$$

Since there was no solution consistent with all the constraints, the vehicle is immobilized by its wheel layout.

Having seen how to compute the motion of abstract, fixed-wheel vehicle layouts, we now turn to a couple of common steering mechanisms that are used to vary the linear and angular velocity of the mobile robot for the purpose of controlling its motion. These steering mechanisms are exactly constrained, so that they will always admit one unique solution.

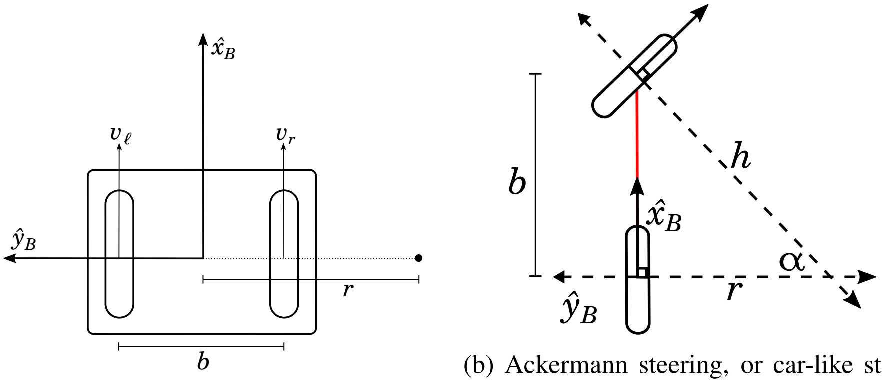
Differential Drive Steering

One can imagine that a two-wheeled variant of the cart would be steerable without slip provided that the two wheels share a common perpendicular axis (Fig. 3.6d). This layout is the differential drive robot steering design. By controlling the velocities of the two wheels (called v_ℓ and v_r), a differential drive mobile robot can achieve an arbitrary linear and angular velocity. The distance between the wheels b must be known. The setup is shown in Fig. 3.8a.

If the individual wheel velocities are known, we can compute the linear and angular velocity of the robot. Noting that the origin of the mobile robot's coordinate frame is located at the center of the two wheels and that $\hat{x}_B$ is parallel to the two wheels' x-axes, the linear velocity of the robot is simply the mean of the wheels' velocities,

$$v = \frac{v_r + v_\ell}{2} \tag{3.35}$$

To compute the angular velocity, we note that whereas linear velocity can vary across particles in a rigid body due to rotation, the angular velocity of the rigid body is the same when measured at any point. Recalling that $\omega r = v$ (a form of Equation 3.18), if we know the velocity at two points, we can use them to find the angular velocity and the radius of curvature. When the robot instantaneously follows a positive radius of curvature r, then it is curving to the left, and so the left wheel (which is on the inside of the curve) follows a radius of curvature of $r - \frac{b}{2}$, whereas the radius of the right wheel (which is on the outside of the curve) is $r + \frac{b}{2}$. Applying the relation,



(a) Differential drive involves a pair of wheels sharing a common orthogonal axis. These robots control their motion by varying the two wheel speeds v_ℓ and v_r individually. The wheels are separated by a distance b, which is assumed known. r is the radius of curvature, measured from the origin of the body frame to the instantaneous center of rotation.

(b) Ackermann steering, or car-like steering, can be approximated by two simple bicycles rigidly affixed. A bicycle is two connected wheels, one of which is fixed whereas the other can steer. They are rigidly joined together and separated by the length of the wheelbase b (red line), assumed known. These robots control their motion by varying the speed of the rear wheel v and the steering angle α of the front wheel.

Figure 3.8: Differential drive and bicycle steering kinematics are shown. The linear velocity v and angular velocity ω are given with respect to the body frame. Both drive mechanisms pictured above have a pair of control inputs that can exactly determine v and ω.

$$\omega\left(r-\frac{b}{2}\right)=v_\ell \tag{3.36}$$

$$\omega\left(r+\frac{b}{2}\right)=v_r. \tag{3.37}$$

Now we can solve for angular velocity:

$$\frac{v_\ell}{\omega}+\frac{b}{2}=r=\frac{v_r}{\omega}-\frac{b}{2} \tag{3.38}$$

$$v_\ell+\omega b=v_r \tag{3.39}$$

$$\omega=\frac{v_r-v_\ell}{b}. \tag{3.40}$$

Finally, we can compute the effective curvature of the differential drive robot's motion,

$$\kappa = \frac{2(v_r - v_\ell)}{b(v_r + v_\ell)}. \tag{3.41}$$

If the desired linear and angular velocites are known, then we can use those to compute desired wheel velocities. By rearranging terms above and then adding or subtracting the equations, we get

$$v_\ell = v - \frac{b\omega}{2} \tag{3.42}$$

$$v_r = v + \frac{b\omega}{2}. \tag{3.43}$$

Ackermann Steering

Ackermann steering, also known as car-like steering, invlves two steerable front wheels and two fixed rear wheels that provide propulsion. Fig. 3.8b illustrates half of an Ackermann steering mechanism for simplicity. A bicycle is controlled through the same principle as a car.

Since the coordinate frame for the bicycle is placed at the rear drive wheel, the velocity of the bicycle is exactly the velocity of the drive wheel. If the linear velocity and steering angle are known, then we can compute the angular velocity by the following reasoning. Recognizing the right triangle formed by the two wheels and the instantaneous rotation center,

$$b = h \sin\alpha \tag{3.44}$$

$$r = h \cos\alpha \tag{3.45}$$

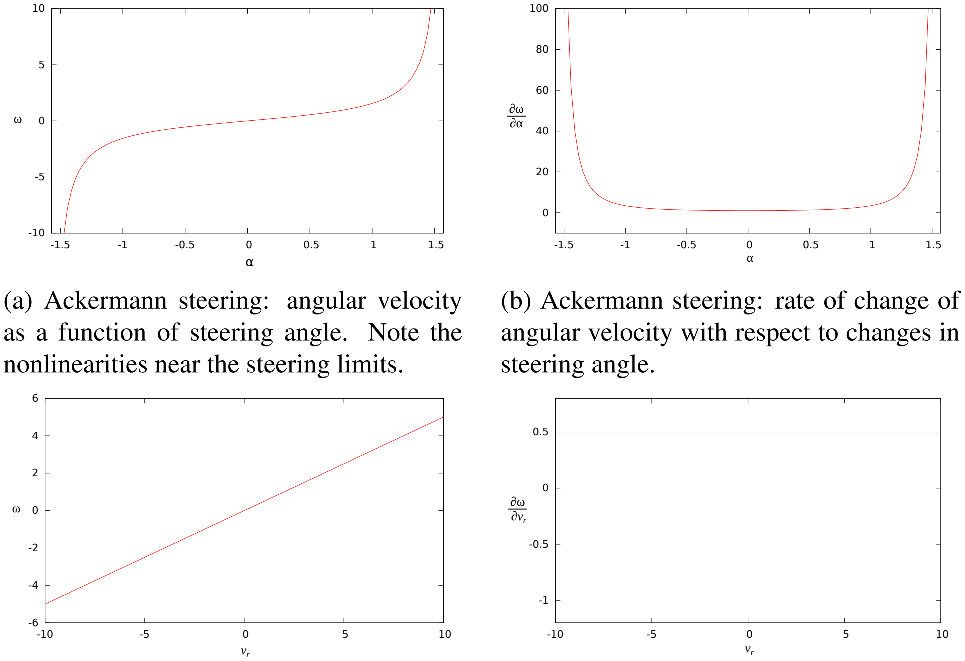
$$\omega = \frac{v}{r} = \frac{v \tan\alpha}{b}. \tag{3.46}$$

Supose now that we are given a desired linear and angular velocity. The corresponding steering angle is computed as

$$\alpha = \text{atan2}(b\omega, v). \tag{3.47}$$

The operator atan2(y, x) is an arctangent that returns angles in the full range from $-\pi$ to π. Note that atan(y/x) only returns angles between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$.

The nonlinearity of (3.47) explains why cars use Ackermann steering rather than differential drive. At high speeds, a car needs to be robust against oversteering, but when maneuvering in a parking lot, a car should be highly responsive to steering input. The variable sensitivity of Ackermann steering to control inputs



(a) Ackermann steering: angular velocity as a function of steering angle. Note the nonlinearities near the steering limits.

(b) Ackermann steering: rate of change of angular velocity with respect to changes in steering angle.

(c) Differential drive: angular velocity as a function of right wheel velocity (for left wheel velocity, multiply by -1).

(d) Differential drive: rate of change of angular velocity with respect to changes in right wheel velocity.

Figure 3.9: The qualitative curves of this figure illustrate why Ackermann steering is used in cars and differential drive is not. Steering inputs to an Ackermann vehicle are highly sensitive to current steering angle, whereas the steering response of a differential drive vehicle is constant.

as a function of current steering angle is key to achieving this high-speed stability and low-speed maneuverability. Fig. 3.9a and Fig. 3.9b illustrate these properties. The relationship shown in Fig. 3.9b is called the *steering response* of the vehicle, which is the rate of change of angular velocity with respect to the control inputs. When the steering wheels of an Ackermann vehicle are near straight, the steering response is minimal. As the steering angle increases at a constant linear velocity, the steering response increases super-linearly.

In contrast, differential drive gives a constant steering across the range of current wheel velocities (see Fig. 3.9c and Fig. 3.9d). Consequently, differential drive tends to make a poor choice of steering model for a car because it is unstable at high speed and frustrating to control in tight spaces because it responds too slowly to changes in the control input. In spite of this, differential drive is often an adequate steering model for a robot, and it has the advantage of being easy to implement.

Regardless of which steering model a vehicle uses, sometimes we lose control of the vehicle — a car might roll, a bicycle might topple over, a wheeled robot might fall off the curb — and our assumption that the vehicle always moves on a smooth, flat surface is violated. When this happens, a 2D description of rigid motion is no longer sufficient. We learned in Section 3.1 that there is one common representation of a 3D translation (a vector that looks just like a position vector). But how do we represent the rotation of a rigid body in three dimensions? The answer is much longer and more complex, and is the subject of the next section.

3.3 Rotation Representations

Many robotics applications require us to describe the rotation of a body in three-dimensional space. For example, suppose that the position of a robot's two-finger end-effector is given by P_A^G. If the robot rotates the revolute joint in its wrist, the position of the end-effector does not change, but the orientation does. Knowing this orientation is key to the robot being able to grasp an object with its fingers rather than knocking the object over or pushing it off the table.

We saw in Section 3.2 that we can describe a 2D rotation by computing the angle between the x-axis of the global frame and the x-axis of the robot's body frame. Suppose that, as in Fig. 3.10, a point A is located on rigid body B. If we say that B's frame $\{B\}$ is rotated by angle θ with respect to global frame $\{G\}$, then we can use this information to compute the global coordinates (position) of A. Remember that the point A does not have an innate orientation, but the act of applying a pure rotation to B causes all points in B to rotate about B's origin, O_B. The operation of rotating A about O_B causes A to translate to a new position.

If we want to compute the new position of A after applying the rotation oper-

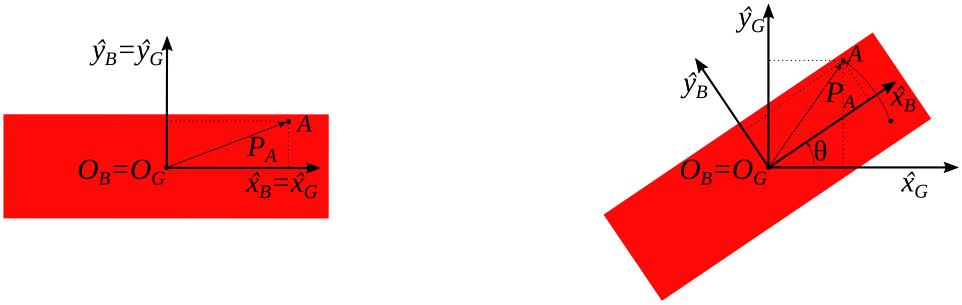


Figure 3.10: At left, a rigid body B before rotation. At right, the same body after being rotated by angle θ. We see that rotating B by θ causes the point A to move to a new position in the global reference frame $\{G\}$, although it is still in the same position in the body frame $\{B\}$.

ation to B, then we need to use some trigonometry. If we know the unchanging coordinates of A in the B frame as $P_A^B = (p_x^B, p_y^B)$, then the transformed coordinates following the rotation are $P_A^G = (p_x^B \cos\theta - p_y^B \sin\theta, p_x^B \sin\theta + p_y^B \cos\theta)$. This result can be derived from inspection, but it may not seem intuitively obvious yet. In the next section, we generalize this result to three dimensions as we begin our study of rotation representations.

3.3.1 The Rotation Matrix

If we take the axes of a body frame $\{B\}$ expressed relative to the global frame $\{G\}$ and use them as the columns of a matrix, the result is a 3×3 matrix called the *rotation matrix*:

$$R_B^G = \begin{bmatrix} \hat{x}_B^G & \hat{y}_B^G & \hat{z}_B^G \end{bmatrix}. \tag{3.48}$$

Since the columns of R_B^G are the axes of $\{B\}$, they are orthogonal unit vectors, and the rotation matrix is therefore an *orthogonal matrix*.

As noted in Section 3.1, the components of a position vector denote the vector's projection onto each axis of a selected reference frame. As a result, each column in the rotation matrix contains the projection of the corresponding axis of $\{B\}$ onto each axis of $\{G\}$. Since the axes are unit vectors, the projection of an axis of $\{B\}$ onto an axis of $\{G\}$ is the cosine of the angle between those two axes (often called a *direction cosine*). Thus,

$$R_B^G = \begin{bmatrix} \cos(\hat{x}_B, \hat{x}_G) & \cos(\hat{y}_B, \hat{x}_G) & \cos(\hat{z}_B, \hat{x}_G) \\ \cos(\hat{x}_B, \hat{y}_G) & \cos(\hat{y}_B, \hat{y}_G) & \cos(\hat{z}_B, \hat{y}_G) \\ \cos(\hat{x}_B, \hat{z}_G) & \cos(\hat{y}_B, \hat{z}_G) & \cos(\hat{z}_B, \hat{z}_G) \end{bmatrix}. \tag{3.49}$$

Alternatively, we can write the direction cosines as dot products, yielding

$$R_B^G = \begin{bmatrix} \hat{x}_B \cdot \hat{x}_G & \hat{y}_B \cdot \hat{x}_G & \hat{z}_B \cdot \hat{x}_G \\ \hat{x}_B \cdot \hat{y}_G & \hat{y}_B \cdot \hat{y}_G & \hat{z}_B \cdot \hat{y}_G \\ \hat{x}_B \cdot \hat{z}_G & \hat{y}_B \cdot \hat{z}_G & \hat{z}_B \cdot \hat{z}_G \end{bmatrix}. \tag{3.50}$$

We can use the fact that the dot product is commutative to derive an interesting property of the rotation matrix. If we transpose R_B^G and then use commutativity to change the order of the vectors in each dot product, we see that

$$(R_B^G)^T = \begin{bmatrix} \hat{x}_G \cdot \hat{x}_B & \hat{y}_G \cdot \hat{x}_B & \hat{z}_G \cdot \hat{x}_B \\ \hat{x}_G \cdot \hat{y}_B & \hat{y}_G \cdot \hat{y}_B & \hat{z}_G \cdot \hat{y}_B \\ \hat{x}_G \cdot \hat{z}_B & \hat{y}_G \cdot \hat{z}_B & \hat{z}_G \cdot \hat{z}_B \end{bmatrix} = R_G^B. \tag{3.51}$$

The transpose of a rotation matrix also has an important relationship with its inverse. If we multiply R_B^G by $(R_B^G)^T$, we see that

$$R_B^G (R_B^G)^T = \begin{bmatrix} \hat{x}_B^G & \hat{y}_B^G & \hat{z}_B^G \end{bmatrix} \begin{bmatrix} (\hat{x}_B^G)^T \\ (\hat{y}_B^G)^T \\ (\hat{z}_B^G)^T \end{bmatrix} = I_3 \tag{3.52}$$

since the columns of R_B^G are mutually orthogonal. Thus,

$$(R_B^G)^T = (R_B^G)^{-1}. \tag{3.53}$$

Another notable property of a rotation matrix R_B^G is that

$$\det(R_B^G) = 1 \tag{3.54}$$

under the assumption of right-handed coordinates. (In general, it can be ± 1.)

Interpretations of the Rotation Matrix

Depending on the application, we may wish to interpret and use the rotation matrix in different ways. Namely, we may use it to

- describe the orientation of a coordinate frame with respect to another frame,
- map the coordinates of a point located in one frame to another frame, and
- rotate a vector to a new orientation in the same coordinate frame.

We provide an example for each of these interpretations below.

Example 3.1 Describing the orientation of a frame

Consider a planar rotation in which a frame is rotated about its z-axis for an angle θ to transition from an initial orientation $\{0\}$ to a final orientation $\{1\}$ (see Fig. 3.11). We would like to describe $\{1\}$ relative to $\{0\}$, so we use (3.49) to derive the expression

$$R_z(\theta) = R_1^0 = \begin{bmatrix} \hat{x}_1^0 & \hat{y}_1^0 & \hat{z}_1^0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.55}$$

where the notation $R_z(\theta)$ is used to denote rotation about the z-axis for an angle θ.

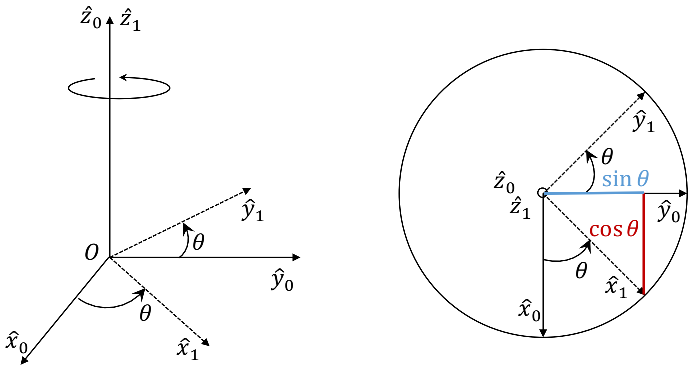


Figure 3.11: Example of a planar rotation about the z-axis from a side (left) and top view.

Similar computations yield the expressions for the rotation matrix describing a rotation about the y-axis,

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{3.56}$$

and the x-axis,

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}. \tag{3.57}$$

Example 3.2 Mapping the coordinates of a point

Assume we are given two frames $\{1\}$ and $\{2\}$ as well as the coordinates of a point A with respect to frame $\{2\}$ in the form of a position vector $P_A^2 = \begin{bmatrix} u & v & w \end{bmatrix}^T$ (see Fig. 3.12). Our goal is to determine the coordinates of A with respect to frame $\{1\}$; that is, we wish to find the position vector P_A^1.

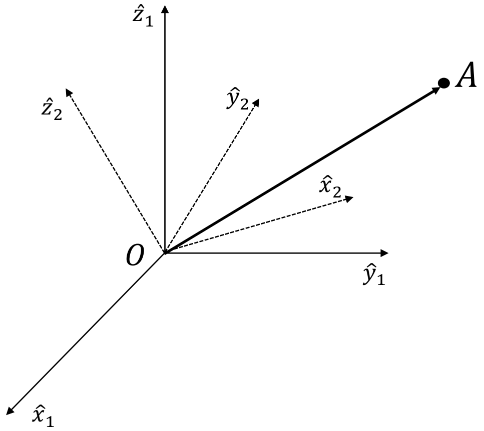


Figure 3.12: The rotation matrix can be used to express a point A with respect to different coordinate frames.

Since the elements of P_A^2 are its projections onto the axes of frame $\{2\}$, we can rewrite P_A^2 as

$$P_A^2 = u\hat{x}_2 + v\hat{y}_2 + w\hat{z}_2. \tag{3.58}$$

We can write a similar expression for P_A^1,

$$P_A^1 = \begin{bmatrix} P_A^2 \cdot \hat{x}_1 \\ P_A^2 \cdot \hat{y}_1 \\ P_A^2 \cdot \hat{z}_1 \end{bmatrix}. \tag{3.59}$$

From equations (3.59) and (3.58), we have

$$\begin{aligned} P_A^1 &= \begin{bmatrix} (u\hat{x}_2 + v\hat{y}_2 + w\hat{z}_2) \cdot \hat{x}_1 \\ (u\hat{x}_2 + v\hat{y}_2 + w\hat{z}_2) \cdot \hat{y}_1 \\ (u\hat{x}_2 + v\hat{y}_2 + w\hat{z}_2) \cdot \hat{z}_1 \end{bmatrix} \\ &= \begin{bmatrix} u\hat{x}_2 \cdot \hat{x}_1 + v\hat{y}_2 \cdot \hat{x}_1 + w\hat{z}_2 \cdot \hat{x}_1 \\ u\hat{x}_2 \cdot \hat{y}_1 + v\hat{y}_2 \cdot \hat{y}_1 + w\hat{z}_2 \cdot \hat{y}_1 \\ u\hat{x}_2 \cdot \hat{z}_1 + v\hat{y}_2 \cdot \hat{z}_1 + w\hat{z}_2 \cdot \hat{z}_1 \end{bmatrix} \\ &= \begin{bmatrix} \hat{x}_2 \cdot \hat{x}_1 & \hat{y}_2 \cdot \hat{x}_1 & \hat{z}_2 \cdot \hat{x}_1 \\ \hat{x}_2 \cdot \hat{y}_1 & \hat{y}_2 \cdot \hat{y}_1 & \hat{z}_2 \cdot \hat{y}_1 \\ \hat{x}_2 \cdot \hat{z}_1 & \hat{y}_2 \cdot \hat{z}_1 & \hat{z}_2 \cdot \hat{z}_1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}. \end{aligned} \tag{3.60}$$

The matrix in the final step of (3.60) is the rotation matrix R_2^1. Thus,

$$P_A^1 = R_2^1 P_A^2. \tag{3.61}$$

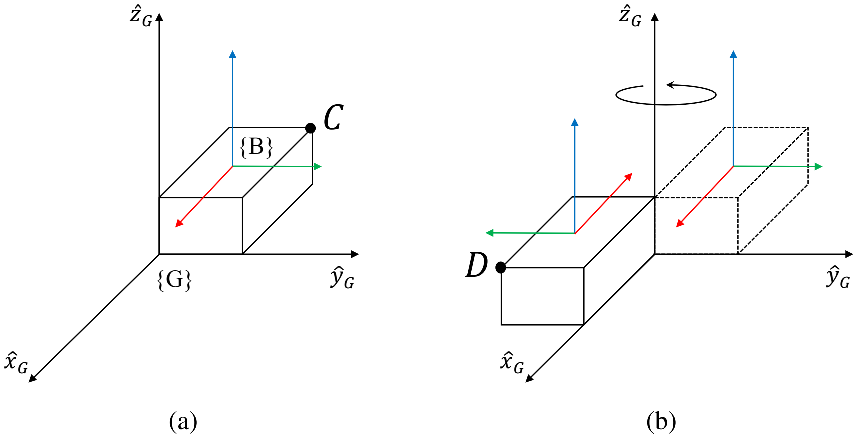


Figure 3.13: A rectangular block starts in the configuration given by (a). After the block is rotated by π radians about $\hat{z}_G$, the new configuration of the block is given by (b).

Equation (3.61) is a useful expression that allows us to map the coordinates of a point from one frame to another.

Example 3.3 Rotating a position vector

Consider the block of Fig. 3.13a in which one corner of the block is located at point C in space. After rotating the block about the axis $\hat{z}_G$ for π radians, the block is configured as shown in Fig. 3.13b, and the same corner of the block is now located at point D. We would like to describe the change in the position of the corner as the rotation of a position vector with respect to the global frame.

To do this, we attach a body frame $\{B\}$ to the block such that $\{B\}$ is coincident with $\{G\}$ before rotation, as shown in Fig. 3.13a. Using $R_z(\theta)$ from (3.55), we can derive the rotation matrix representing the final orientation of $\{B\}$ with respect to the global frame $\{G\}$, as

$$R_B^G = R_z(\pi) = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.62}$$

Now we can use (3.61) from the previous example to relate the coordinates of D within the body frame to the coordinates of D within the global frame, as

$$P_D^G = R_B^G P_D^B. \tag{3.63}$$

But since the position of the corner never changes with resepct to $\{B\}$, it must be the case that $P_D^B = P_C^G$ when $\{B\}$ and $\{G\}$ are aligned as in Fig. 3.13a. We can therefore substitute P_C^G into (3.63) to obtain

$$P_D^G = R_B^G P_C^G. \tag{3.64}$$

Equation (3.64) thus allows us to express the rotational motion of a vector within the same coordinate frame.

Composition of Rotations

Articulated robotic mechanisms are composed of several rigid bodies attached to each other by joints of various kinds. As a result, to describe their kinematics, it is often necessary to compose the motions of their components. The typical approach to this problem is to attach a frame to each rigid body, determine the relative displacements between successive bodies, and combine those displacements into a net displacement. Importantly, when composing a sequence of rotational displacements, the method of composition depends on whether the rotations were made with respect to the *current frame* or to the *fixed frame*. To understand the difference between these types of rotations, we will consider the following example.

Suppose that a manipulator arm has three links and two revolute joints rotating about orthogonal axes. We label the links incrementally starting with the link attached to the base, which we call link 0. We then attach a frame to each link, again starting from the base, and call the frames $\{0\}$, $\{1\}$, and $\{2\}$. For the moment, let us assume that the displacement from one link to the next is a pure rotation about a common origin. Suppose that the position of point A in frame $\{2\}$ is given by P_A^2. To determine the position of A relative to frame $\{0\}$, we can use the following two relationships:

$$P_A^0 = R_1^0 P_A^1, \tag{3.65}$$

$$P_A^1 = R_2^1 P_A^2. \tag{3.66}$$

Substituting (3.66) into (3.65) yields

$$P_A^0 = R_1^0 R_2^1 P_A^2. \tag{3.67}$$

What this equation says is that in order to transform the coordinates of a point A from frame $\{2\}$ to frame $\{0\}$, we may first transform P_A^2 to P_A^1 using R_2^1 and then transform P_A^1 to P_A^0 using R_1^0. Furthermore, since we know that

$$P_A^0 = R_2^0 P_A^2, \tag{3.68}$$

we can infer from (3.68) and (3.67) that

$$R_2^0 = R_1^0 R_2^1. \tag{3.69}$$

We can interpret (3.69) in the context of our three-link, two-joint arm. As we move up the arm from the base, each joint applies a rotation matrix (which is a function of the joint angle) to the previous frame in the chain to compute the new frame. Thus, to compute the orientation of $\{2\}$ with respect to $\{0\}$, we first apply R_1^0 to move up the chain from $\{0\}$ to $\{1\}$, and then we apply R_2^1 to move up the chain from $\{1\}$ to $\{2\}$. Note that the frame to which the rotation matrix is relative changes from $\{0\}$ to $\{1\}$ during this process. We call the frame that is relative to the rotation we are currently performing the **current frame**.

Sometimes we may wish to perform rotations with respect to a single, non-changing frame rather than the current frame. When we do this, we refer to the single frame as the **fixed frame**. Suppose that we select $\{0\}$ to be the fixed frame. In this case, R_1^0 is a rotation with respect to the fixed frame, but R_2^1 is not. We therefore define another rotation matrix R that has the same value as R_2^1 but is named differently to reflect that it is a rotation with respect to the fixed frame rather than the current frame. The interpretation is not as nice as just moving up the chain, but we can try to visualize it in the following way:

1. First, we apply R_1^0 to move up the chain from $\{0\}$ to $\{1\}$. When applying the first rotation, it is simultaneously in the current and fixed frames.

2. Since we are at the current frame rather than the fixed frame, we need to apply another transformation before we can use R. We therefore apply $(R_1^0)^{-1} = R_0^1$ in the current frame to move back to the fixed frame.

3. Now that we are at the fixed frame, we apply R in the current frame (which is also the fixed frame).

4. Finally, we need to undo the transformation we applied in the second step, so we apply the transformation $(R_0^1)^{-1} = R_1^0$ in the current frame.

Putting these steps together yields

$$R_2^0 = R_1^0 R_0^1 R R_1^0. \tag{3.70}$$

You may have noticed that the first two steps seem redundant: we move from the fixed frame to $\{1\}$ and then right back to the fixed frame. Indeed, since R_0^1 is the inverse of R_1^0, we have $R_1^0 R_0^1 = I_3$. Thus, our equation is reduced to

$$R_2^0 = R R_1^0. \tag{3.71}$$

It is illuminating to compare (3.71) side-by-side with (3.69). To obtain R_2^0 by performing a rotation R with respect to the **current frame**, we **postmultiplied** R_1^0 by $R = R_2^1$ to obtain

$$R_2^0 = R_1^0 R_2^1. \tag{3.72}$$

In contrast, when we performed the second rotation with respect to the **fixed frame**, we **premultiplied** R_1^0 by R to obtain

$$R_2^0 = RR_1^0. \tag{3.73}$$

Why do we draw the distinction between fixed frame and current frame? It matters when and how new rotations are applied. For example, given a robot arm in a known configuration, bending the shoulder joint incurs a rotation in the fixed frame because the shoulder occurs *earlier* in the arm than the other joints. Similarly, a bending of the wrist creates a rotation in the current frame because it is *later* in the arm than the other joints. Note that in general, the R_2^0 in (3.72) and the R_2^0 in (3.73) will not be the same because matrix multiplication does not commute.

It should be noted that, given a product of arbitrary rotations $R_aR_bR_cR_d$, one can always achieve the same mathematical result by applying rotations in either the current frame or fixed frame, or any mix of the two. For example, these are both valid ways to achieve the same net rotational displacement of the final frame with respect to the unrotated frame:

- Rotate by R_b in the current frame: $IR_b = R_b$
- Rotate by R_c in the current frame: R_bR_c
- Rotate by R_a in the fixed frame: $R_aR_bR_c$
- Rotate by R_d in the current frame: $R_aR_bR_cR_d$

or

- Rotate by R_c in the fixed frame: $R_cI = R_c$
- Rotate by R_d in the current frame: R_cR_d
- Rotate by R_b in the fixed frame: $R_bR_cR_d$
- Rotate by R_a in the fixed frame: $R_aR_bR_cR_d$

To avoid ambiguity when composing rotations, one must always specify whether each rotation is being applied in the current or fixed frames.

Beyond the Rotation Matrix

We previously alluded to the fact that there are many ways to represent rotations in three dimensions. What might a representation other than the rotation matrix look like? One might wonder whether we can represent a 3D rotation with fewer elements than the nine elements of the rotation matrix. The answer is that we can, and we can actually use the rotation matrix itself to determine the minimum number of elements required. Recall that we originally introduced the rotation matrix as

$$R_B^G = \begin{bmatrix} \hat{x}_B^G & \hat{y}_B^G & \hat{z}_B^G \end{bmatrix} \tag{3.74}$$

where the columns of R_B^G are orthogonal unit vectors. The fact that they are unit vectors gives us the dependencies

$$\left\|\hat{x}_B^G\right\| = 1, \qquad \left\|\hat{y}_B^G\right\| = 1, \qquad \left\|\hat{z}_B^G\right\| = 1, \tag{3.75}$$

and the fact that they are mutually orthogonal gives us the dependencies

$$\hat{x}_B^G \cdot \hat{y}_B^G = 0, \qquad \hat{x}_B^G \cdot \hat{z}_B^G = 0, \qquad \hat{y}_B^G \cdot \hat{z}_B^G = 0. \tag{3.76}$$

Thus we have six equations that constrain the nine elements of the rotation matrix. As a result, only three of those nine elements are independent quantities, which implies that a rigid body possesses at most three rotational degrees of freedom. It follows that any 3D rotation can be expressed with just three parameters. Several rotation representations make use of this fact, and in the following sections we present several notable ones: Euler angles, the axis-angle pair, and quaternions.

3.3.2 Euler Angles

Since a rigid body has up to three rotational degrees of freedom, any orientation can be achieved by three rotations about the axes of a frame. The **Euler angles** rotation representation describes three such rotations. Whenever Euler angles are used, it is crucial to specify (1) the order in which the rotations are applied, (2) whether they are applied in the fixed frame or the current frame, and (3) which axes are being rotated about. We will denote Euler angles as α, β, and γ, but different authors use different names for the angles, so the names chosen for the angles should not be read into too much. What is most important is the sequence in which they occur.

The axes that we choose to rotate about and the order in which the rotations are applied define an Euler angle *convention*, each with a specific name. For example, if the first rotation is about the x-axis, the second is about the y-axis, and the third is about the z-axis, then we call our angles X-Y-Z Euler angles. With some thought we can conclude that there are twelve possible sequences. Six of these sequences result from all possible permutations of X, Y, and Z:

X-Y-Z, X-Z-Y, Y-X-Z, Y-Z-X, Z-X-Y, Z-Y-X.

The other six are all possible sequences where the first and third rotation axes are the same and the second is different:

X-Y-X, X-Z-X, Y-X-Y, Y-Z-Y, Z-X-Z, Z-Y-Z.

Note that the convention names, when written this way, still do not tell us whether the rotations are about the fixed frame or the current frame. For this reason, some authors will use the names written above to denote rotations about the fixed frame,

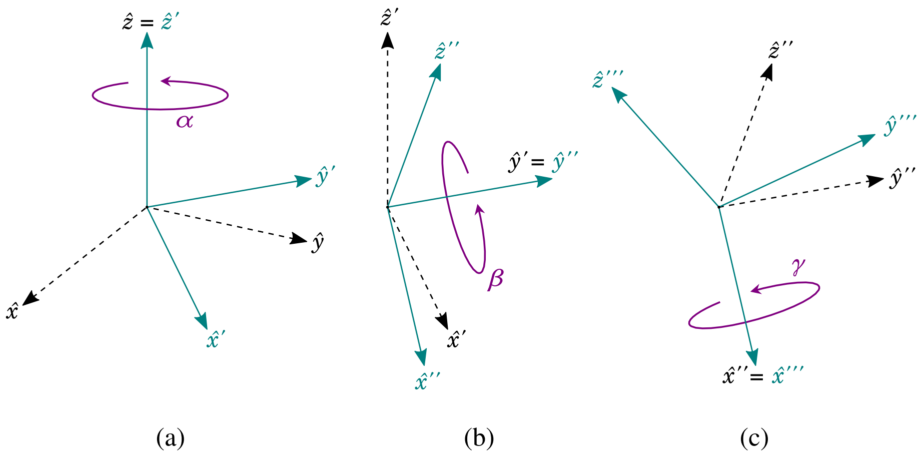


Figure 3.14: These Z-Y′-X″ Euler angles, denoted (α, β, γ), represent the orientation of the frame given by the axes $\hat{x}'''$, $\hat{y}'''$, $\hat{z}'''$ with respect to the frame given by the axes $\hat{x}$, $\hat{y}$, $\hat{z}$. Note that all the rotations were applied in the current frame.

and use names such as Z-Y′-X″ to denote rotations about the current frame (see Fig. 3.14 for an illustration of this convention).

The rotations shown in Fig. 3.14 can be represented as rotation matrices $R_z(\alpha)$, $R_{y'}(\beta)$, $R_{x''}(\gamma)$. They can be composed into a rotation matrix $R_{ZY'X''}$ as follows:

$$R_{ZY'X''} = R_z(\alpha)R_{y'}(\beta)R_{x''}(\gamma) \tag{3.77}$$

$$= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{bmatrix} \tag{3.78}$$

$$= \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \tag{3.79}$$

where c_{angle} and s_{angle} denote $\cos(\text{angle})$ and $\sin(\text{angle})$ respectively. The rotation matrix $R_{ZY'X''}$ is equivalent to the $ZY'X''$ Euler angles.

Since we are calling these representations “equivalent”, one might wonder why we would ever prefer the more complicated rotation matrix representation over the Euler angle representation. It turns out that using only three parameters to represent a 3D rotation has its drawbacks, and these drawbacks can make Euler angles unsuitable for some practical applications. Let us suppose that we have a set of $ZY'X''$ Euler angles given by $(\alpha = \pi/4,\ \beta = \pi/2,\ \gamma = \pi/4)$. After the second rotation (90 degrees about $\hat{y}'$) is applied, $\hat{x}'$ is aligned with $\hat{z}''$. The fact that these axes are aligned causes our 3D representation to lose a degree of freedom.

What does this mean mathematically? Let's plug $\beta = \pi/2$ into (3.79) and find out:

$$R_{ZY'X''} = \begin{bmatrix} c_\alpha \cdot 0 & c_\alpha \cdot 1 \cdot s_\gamma - s_\alpha c_\gamma & c_\alpha \cdot 1 \cdot c_\gamma + s_\alpha s_\gamma \\ s_\alpha \cdot 0 & s_\alpha \cdot 1 \cdot s_\gamma + c_\alpha c_\gamma & s_\alpha \cdot 1 \cdot c_\gamma - c_\alpha s_\gamma \\ -1 & 0 \cdot s_\gamma & 0 \cdot c_\gamma \end{bmatrix} \tag{3.80}$$

$$= \begin{bmatrix} 0 & c_\alpha s_\gamma - s_\alpha c_\gamma & c_\alpha c_\gamma + s_\alpha s_\gamma \\ 0 & s_\alpha s_\gamma + c_\alpha c_\gamma & s_\alpha c_\gamma - c_\alpha s_\gamma \\ -1 & 0 & 0 \end{bmatrix} \tag{3.81}$$

$$= \begin{bmatrix} 0 & -s_{\alpha-\gamma} & c_{\alpha-\gamma} \\ 0 & c_{\alpha-\gamma} & s_{\alpha-\gamma} \\ -1 & 0 & 0 \end{bmatrix} \tag{3.82}$$

What we can conclude is that when $\beta = \pi/2$, changing the values of α and γ will change the rotation angle $\alpha - \gamma$, but the rotation axis, which is $\hat{z}$, will never change because the first column and last row of the rotation matrix will never change. Thus, α and γ have the same role, and as a result have lost a degree of freedom. The only way to recover all three degrees of freedom is to change β. In practical applications, this effect is often called *gimbal lock*. We call a point at which gimbal lock occurs a *singularity*. Every Euler angle convention has two singularities. For the six conventions that are a permutation of X, Y, and Z, the singularities occur at $\beta = \pi/2$ and $\beta = 3\pi/2$ (assuming β is the rotation that is applied second). For the six conventions in which the first and third rotation axes are the same, the singularities occur at $\beta = 0$ and $\beta = \pi$, again assuming β is the second rotation.

Thus, although Euler angles are very intuitive, they suffer from the gimbal lock problem no matter what convention you use. We will now turn our attention to other rotation representations that avoid this problem.

3.3.3 Axis-Angle Representation

In addition to giving us Euler angles, Leonhard Euler also gave us the **axis-angle** representation by proving that any two coordinate frames with a common origin are related by a single rotation about some fixed axis. As a result, we can represent any 3D rotation with one unit vector $\hat{x}$ and one angle θ (see Fig. 3.15).

You may recall that we have already seen some axis-angle pairs. If $\hat{k}$ is selected to be one of the principle axes, then we have one of the following basic rotations:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \tag{3.83}$$

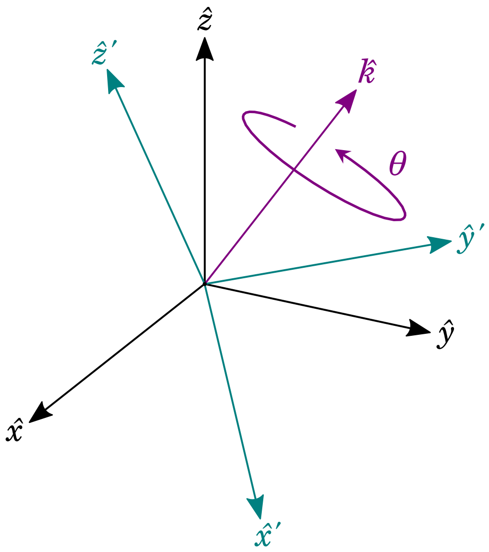


Figure 3.15: Axis-angle representation is given by an axis of rotation $\hat{k}$ and an angle θ.

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{3.84}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.85}$$

Where things become more interesting is when $\hat{k}$ is an arbitrary axis. In this case, the rotation matrix corresponding to the axis-angle pair $R_k(\theta)$ is more complicated:

$$R_k(\theta) = \begin{bmatrix} k_x^2(1-c_\theta)+c_\theta & k_xk_y(1-c_\theta)-k_zs_\theta & k_xk_z(1-c_\theta)+k_ys_\theta \\ k_yk_x(1-c_\theta)+k_zs_\theta & k_y^2(1-c_\theta)+c_\theta & k_yk_z(1-c_\theta)-k_xs_\theta \\ k_zk_x(1-c_\theta)-k_ys_\theta & k_zk_y(1-c_\theta)+k_xs_\theta & k_z^2(1-c_\theta)+c_\theta \end{bmatrix}. \tag{3.86}$$

We can also consider the inverse problem: given an arbitrary rotation matrix, can we derive the corresponding axis-angle pair? There is also a straightforward way to do this. Given some rotation matrix

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{3.87}$$

the angle θ of the equivalent axis-angle pair is given by

$$\theta = \cos^{-1}\left(\frac{r_{11}+r_{22}+r_{33}-1}{2}\right). \tag{3.88}$$

and the axis $\hat{k}$ is given by

$$\hat{k} = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix}, \tag{3.89}$$

Since the computation of the angle θ requires an arccosine, the value of θ is always between 0 and π radians. What this suggests is that there is not one unique axis-angle pair that corresponds to any given rotation matrix. Indeed, a rotation of θ radians about $\hat{k}$ is equivalent to a rotation of $-\theta$ radians about $-\hat{k}$. We therefore need to choose between these two solutions when converting a rotation matrix to an axis-angle pair. In addition, we need to be careful that we do not try to plug in $\theta = 0$ or $\theta = \pi$ to our formula for the axis since that will result in division by zero. If $\theta = 0$, then there is no rotation, and so the axis of rotation is undefined. On the other hand, if $\theta = \pi$, then there is an axis of rotation, but the sign is ambiguous.

As we have described it, the axis-angle representation requires four real numbers: a three-vector axis and a scalar angle. However, since the axis is of unit length, only two of its components are independent. We can therefore compress the axis-angle representation into a single three-vector, as $v = \begin{bmatrix}\theta k_x & \theta k_y & \theta k_z\end{bmatrix}^T$. Then the length of v is the angle θ and the direction of v is the axis $\hat{k}$. It is important to note that although we can represent an axis-angle pair with a single vector, we cannot use the standard rules of vector algebra to compose rotations. If we could, it would imply that rotations are commutative, which in general is false.

3.3.4 Quaternions

We have seen that the axis-angle rotation allows us to express a 3D rotation with four real numbers, but when we express those four numbers as a single vector, we cannot use standard vector algebra to work with them. One might ask: is there another kind of algebra we can use? Asking this question leads us to the final 3D rotation representation we will study: **quaternions**. A quaternion can be thought of as a complex number with three imaginary parts, a scalar and a three-vector, or a four-vector. These three different ways of thinking about quaternions lead to three common notations:

$$Q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}, \tag{3.90}$$

where $q_0, q_1, q_2, q_3 \in \mathbb{R}$ and **i**, **j** and **k** are the fundamental unit quaternions,

$$Q = (q_0, \vec{q}), \tag{3.91}$$

where q_0 is the *scalar component* and $\vec{q} \in \mathbb{R}^3$ is the *vector component*, and

$$Q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T. \tag{3.92}$$

The class of quaternions that allows us to represent rotations are the *unit quaternions*, which is the set of quaternions such that $\|Q\| = 1$. As stated above, **i**, **j** and **k** are the fundamental unit quaternions and can be thought of as unit vectors pointing along the spacial axes (akin to $\hat{x}$, $\hat{y}$, and $\hat{z}$). In general, a rotation of θ radians about a unit vector $\hat{k}$ can be represented by the unit quaternion

$$Q = \begin{bmatrix} \cos\frac{\theta}{2} & k_x \sin\frac{\theta}{2} & k_y \sin\frac{\theta}{2} & k_z \sin\frac{\theta}{2} \end{bmatrix}^T. \tag{3.93}$$

We may also recover the axis-angle pair given the unit quaternion, as

$$\theta = 2\cos^{-1} q_0 \quad \text{and} \quad \hat{k} = \frac{\vec{q}}{\sqrt{1 - q_0^2}}. \tag{3.94}$$

Note that if $q_0 = 1$, then $\theta = 0$, and our formula for computing $\hat{k}$ breaks because the axis of rotation is undefined.

In addition to converting between quaternions and axis-angle representation, we can convert between quaternions and rotation matrices. Given a rotation matrix R, the scalar and vector components of the corresponding quaternion are given by

$$q_0 = \frac{1}{2}(1 + r_{11} + r_{22} + r_{33})^{1/2}, \tag{3.95}$$

$$\vec{q} = \frac{1}{4q_0} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}. \tag{3.96}$$

Conversely, given a unit quaternion, the corresponding rotation matrix is

$$R = \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix}. \tag{3.97}$$

We do not need to convert quaternions to rotation matrices before we can use them, however; we can use quaternion multiplication directly. Importantly, quaternion multiplication is distributive and associative but *not* commutative. For any two quaternions Q and N, the product $Q \cdot N$ is given by

$$Q \cdot N = \begin{bmatrix} q_0 n_0 - q_1 n_1 - q_2 n_2 - q_3 n_3 \\ q_0 n_1 + q_1 n_0 + q_2 n_3 - q_3 n_2 \\ q_0 n_2 + q_2 n_0 + q_3 n_1 - q_1 n_3 \\ q_0 n_3 + q_3 n_0 + q_1 n_2 - q_2 n_1 \end{bmatrix}. \tag{3.98}$$

To compose consecutive rotations, we can multiply their corresponding quaternions like we did with rotation matrices. Suppose we have two quaternions Q_1^0 and Q_2^1 which represent the orientation of frame $\{1\}$ relative to frame $\{0\}$ and the orientation of frame $\{2\}$ relative to frame $\{1\}$ respectively. Using quaternion multiplication, we may derive the quaternion expressing the orientation of frame $\{2\}$ with respect to frame $\{0\}$ as

$$Q_2^0 = Q_1^0 \cdot Q_2^1. \tag{3.99}$$

We also may wish to apply a rotation to a position vector using quaternions. To do this, we first need to define the *conjugate* of a quaternion $Q = (q_0, \vec{q})$. The conjugate of Q is given by $Q^* = (q_0, -\vec{q})$ and satisfies the following equation:

$$\|Q\|^2 = Q \cdot Q^* = q_0^2 + q_1^2 + q_2^2 + q_3^2. \tag{3.100}$$

We can now define the *inverse* of a quaternion, as

$$Q^{-1} = \frac{Q^*}{\|Q\|^2}. \tag{3.101}$$

Now suppose that we have some position vector $P_A = \begin{bmatrix} x & y & z \end{bmatrix}^T$. To rotate P_A using the quaternion Q, we perform quaternion multiplication like so:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = Q \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \cdot Q^{-1}. \tag{3.102}$$

Then all we need to do is use x', y', and z' to form the newly-rotated position vector, as $P'_A = \begin{bmatrix} x' & y' & z' \end{bmatrix}^T$.

3.4 Properties of Rotations

Now that we have seen several ways to represent a 3D rotation, one may wonder: when do we select one parameterization over another, and why? In this section, we discuss some of the properties that we need from any representation of rotations, and we consider the properties of various representations that may be useful or problematic in an implementation.

Some of the most important properties when writing software that computes rotations for a robot are as follows:

- **Dimensional generality.** We prefer that representations support rotations in both two and three dimensions.
- **Compactness.** As a rule, we like data structures to be no larger than they need to be. If a representation takes more than the minimal amount of space required to store information, we say that it has redundancy. Algorithms compute faster on smaller data structures because more of them fit into memory cache. They also transmit more quickly over networks than larger data structures. With modern computers, we can afford to pay less attention to this property. However, it is related to another similar property: uniqueness.
- **Uniqueness.** We prefer that for each possible rotation we might want to express, there should be one unique representational form of that rotation. Having redundant representations of the same rotation suggests that the data structure could be made more compact, and also that comparisons for equality are more difficult.
- **Numerical stability.** Computations with floating point numbers are subject to numerical error. For example, with repeated multiplications, numbers may lose precision in subtle ways. If a representation has redundancy, then some possible values in the representation may not correspond to any physical rotation. A good representation should be numerically stable; that is, it should degrade gracefully in the face of accumulating numerical error.
- **Interpolability.** We may wish to interpolate from one orientation to another, with each orientation given by some parameterization of a rotation. The naive approach to interpolation with a scalar, vector, or matrix is to linearly interpolate each value based on a parameter $t \in [0, 1]$. Interpolation is not trivial, however, when a representation has redundancy since intermediate values under a linear interpolation do not in general correspond to physically meaningful orientations. Even with representations for which linear interpolation produces valid rotations, the result of linear interpolation may be counterintuitive.
- **Differentiability.** Besides composing discrete rotations, we may wish to describe rotational rates, in which case we would like to have the property that for any orientation, a given rate of change of that orientation should correspond at least roughly to a constant rate of change in the parameterization.

The properties above are the metrics by which we should judge each representation for use in robotics applications. The table below shows how the various representations defined in Section 3.3 compare to each other according to these metrics.

Representation	Generality	Compactness	Uniqueness	Numerical Stability	Interpolability	Differentiability
angle	2D	Y	N	Y	Y	Y*
rotation matrix	Y	N	Y	N	N	Y
Euler angles	3D	Y	Y*	Y	N	N
axis-angle	3D	Y	Y	Y	N†	N†
quaternion	3D	N	N‡	N	Y	Y

* true in general, but fails at a finite number of discrete parameter values.

† except when the interpolation or differentiation is about the axis of rotation.

‡ the set of unit quaternions is a double-cover of SO(3); that is, for each orientation, there is a corresponding quaternion with both positive and negative real value.

3.4.1 Algebraic Properties of Rotations

An *algebraic group* is a set of mathematical objects that all share an operator. Rotations define a group under the *composition operator*, where composition means performing one rotation followed by another rotation. In two dimensions, the composition operator for angles is addition modulo 2π, whereas for rotation matrices the composition operator is multiplication.

The rotation group under composition is called SO(2) in two dimensions and SO(3) in three dimensions. "SO" stands for the special orthogonal group. As with all groups, SO(2) and SO(3) have four properties:

- **Closure.** The composition of any two rotations is itself a valid rotation.
- **Identity.** There exists an element of the set of rotations, I, that when composed with any other rotation R yields R. Thus, $IR = RI = R$.
- **Inverse.** For every element R in the set of rotations, there exists an inverse element R^{-1} such that when composed with R, the result is the identity. Thus, $RR^{-1} = R^{-1}R = I$.
- **Associativity.** When composing three rotations, we can simplify either pair first and get the same result. Thus, $ABC = (AB)C = A(BC)$.

One property that SO(2) and SO(3) do not share is commutativity (i.e. for all A, B in the group, $AB = BA$). We call a group in which commutativity holds for the group operator an *Abelian group*. As a general rule, rotation does not commute, and so SO(3) is not Abelian. In other words, the order in which you apply rotations in 3D generally gives a different result.

In contrast, SO(2) *is* Abelian because real numbers (angles) commute over addition modulo 2π. As a sanity check, we can confirm that 2D rotation matrices should also commute under multiplication. This is somewhat surprising because in general matrix multiplication does not commute, but interestingly the structure of 2D rotation matrices is special in a way that causes them to commute. We can see that this is true as follows:

$$R(\alpha)R(\beta) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} \tag{3.103}$$

$$= \begin{bmatrix} \cos\alpha\cos\beta - \sin\alpha\sin\beta & -\cos\alpha\sin\beta - \sin\alpha\cos\beta \\ \cos\alpha\sin\beta + \sin\alpha\cos\beta & \cos\alpha\cos\beta - \sin\alpha\sin\beta \end{bmatrix} \tag{3.104}$$

$$R(\beta)R(\alpha) = \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{3.105}$$

$$= \begin{bmatrix} \cos\alpha\cos\beta - \sin\alpha\sin\beta & -\cos\alpha\sin\beta - \sin\alpha\cos\beta \\ \cos\alpha\sin\beta + \sin\alpha\cos\beta & \cos\alpha\cos\beta - \sin\alpha\sin\beta \end{bmatrix} \tag{3.106}$$

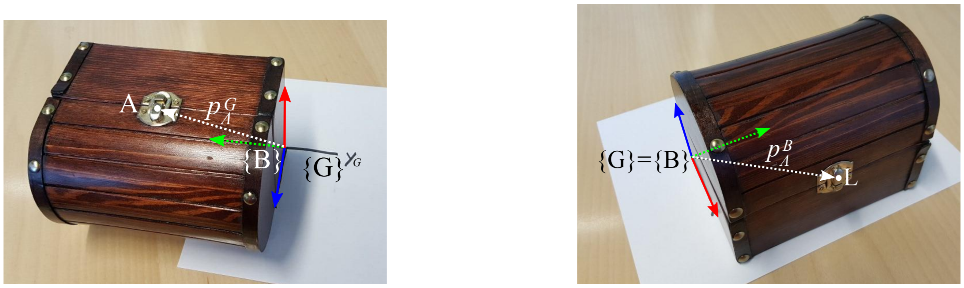
Although SO(3) is not Abelian, commutativity does apply to rotation matrices if they rotate about the same axis. For example, $R_X(\alpha)R_X(\theta) = R_X(\theta)R_X(\alpha)$.

3.4.2 Long-Form Example

Example 3.4 Fixed Frame vs. Current Frame Euler Angles

Concepts reviewed: *expressing coordinates in a new frame*, *coordinate transformation*, *composing rotations*, *Euler angles*, *body frame vs. global frame*, *current frame vs. fixed frame*.

Problem: Suppose you want to open the treasure chest in Fig. 3.16a lying on its back with the latch at point A identified by coordinates P_A^B or P_A^G. You know the coordinates P_L^G of a point L which corresponds to the latch on the unrotated chest in Fig. 3.16b. You want to open the latch and reveal its treasure, but your robot is in the global frame $\{G\}$, and it cannot directly reach to the point A without transforming it into the $\{G\}$ frame first. You are given the orientation of the box with Euler angles $\alpha = \frac{\pi}{2}, \beta = \frac{\pi}{2}, \gamma = \frac{\pi}{2}$ using the Euler angle convention R_{XYZ}, but you do not know whether it is current or fixed frame. Determine which one is correct and find P_A^G.



(a) Rotated body frame

(b) Unrotated global frame

Figure 3.16: Problem statement: a treasure chest is configured as shown in (a), but the position of the latch at point A is only known in the global frame as in (b). Find P_A^B. The red, green, and blue axes represent the coordinate frame $\{B\}$ of the body of the treasure chest, and they are fixed to its three straight edges. The global frame $\{G\}$ is on the paper. In (b), the two frames are aligned.

Solution: We compute the Euler angle twice, as both fixed-frame and current-frame, and check which one fits the final orientation.

Note that this problem can be interpreted in two ways. First, we could interpret the problem as *expressing coordinates in a new frame*. We note that $P_A^B = P_L^B$ because the chest is a rigid body and points L and A both correspond to the same latch. Furthermore, $P_L^B = P_L^G$ because L is defined when the body's coordinate frame is aligned with the global one. Thus, we know P_A^B and merely need to express it in the global coordinate frame as $P_A^G = R_B^G P_A^B$.

Alternatively, we could interpret the problem as a *coordinate transformation* by recognizing that at some point in the past, the chest was rotated from its canonical configuration to the way we found it. At that time, the latch moved from the point L to the point A. Therefore, $P_A^G = R_B^G P_L^G$.

Fixed Frame. To perform a rotation in the fixed frame (which in our case is the global frame), we premultiply. Thus,

$$R_B^G = R_{XYZ} = R_Z(\gamma) R_Y(\beta) R_X(\alpha) \tag{3.107}$$

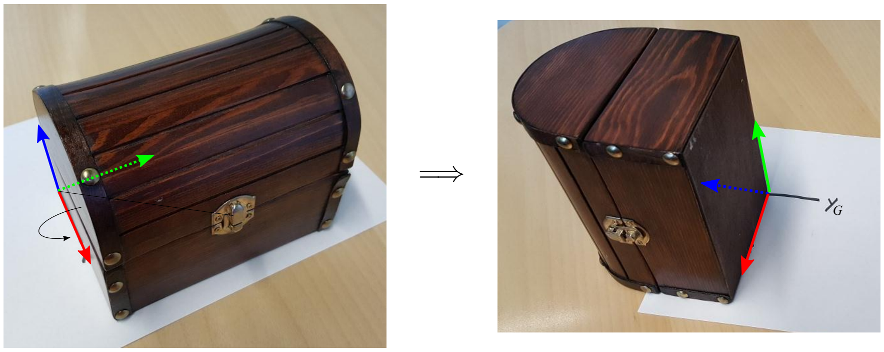
$$= \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha \\ 0 & s_\alpha & c_\alpha \end{bmatrix}. \tag{3.108}$$

Let us consider these steps one at a time.

1. Premultiply

$$R_X(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha \\ 0 & s_\alpha & c_\alpha \end{bmatrix} \tag{3.109}$$

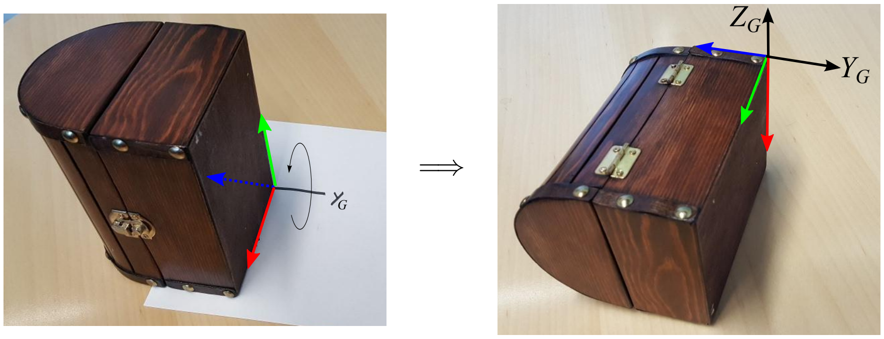
This rotates about the global $+X$ axis, which is also the body $+X$ axis.



2. Premultiply

$$R_Y(\beta) = \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \tag{3.110}$$

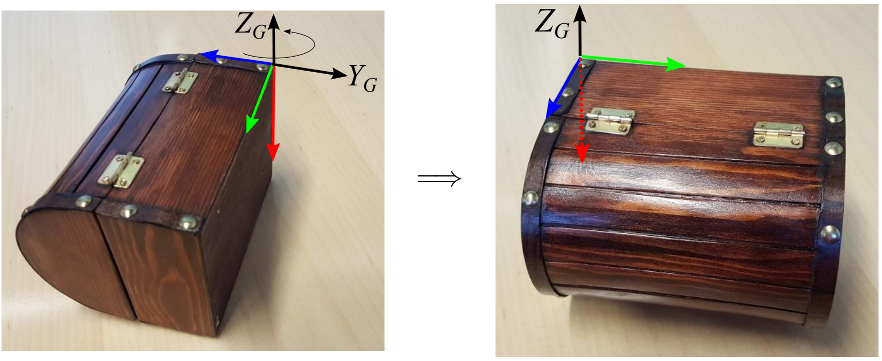
This rotates about the global $+Y$ axis, which is also the body $-Z$ axis.



3. Premultiply

$$R_Z(\gamma) = \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.111}$$

This rotates about the global $+Z$ axis, which is also the body $-X$ axis.



$$\begin{aligned}
P_A^G =& R_B^G P_A^B = R_{XYZ} P_A^B && (3.112)\\
=& \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha \\ 0 & s_\alpha & c_\alpha \end{bmatrix} P_A^B && (3.113)\\
=& \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} P_A^B && (3.114)\\
=& \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} P_A^B. && (3.115)
\end{aligned}$$

One can visually verify the computed rotation matrix by noting that its three column vectors match the three body-frame coordinate axes when expressed in the global frame. This configuration does not match the desired final configuration. □

Current Frame. Rotations in the current frame are accomplished by postmultiplying. Thus,

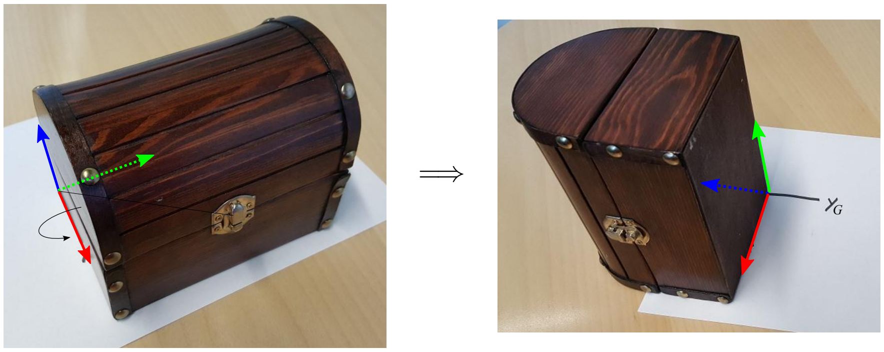
$$\begin{aligned}
R_B^G =& R_{XYZ} = R_X(\alpha) R_Y(\beta) R_Z(\gamma) && (3.116)\\
=& \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha \\ 0 & s_\alpha & c_\alpha \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. && (3.117)
\end{aligned}$$

Let us again consider these steps one at a time.

1. Postmultiply

$$R_X(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha \\ 0 & s_\alpha & c_\alpha \end{bmatrix} \qquad (3.118)$$

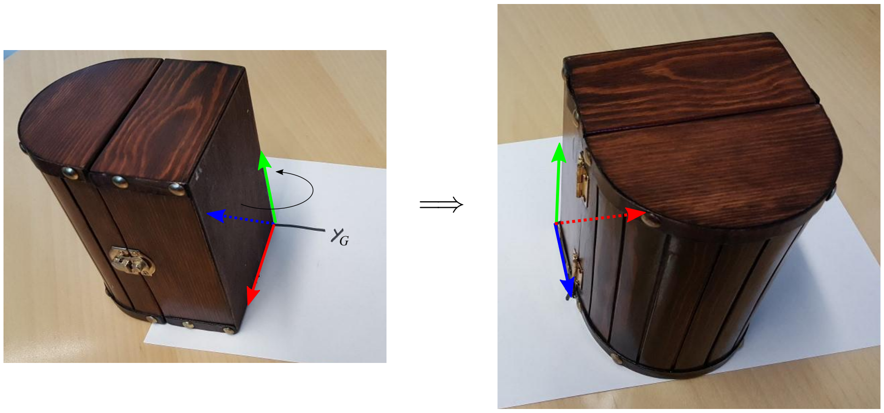
This rotates about the body $+X$ axis, which is also the global $+X$ axis.



2. Postmultiply

$$R_Y(\beta) = \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \tag{3.119}$$

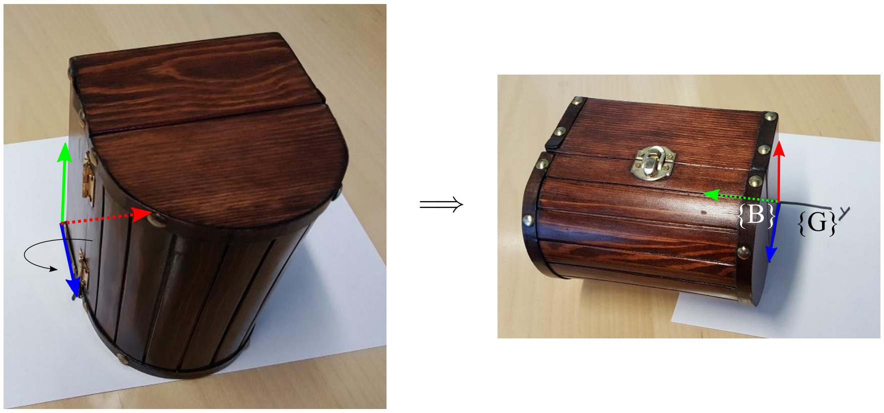
This rotates about the body $+Y$ axis, which is also the global $+Z$ axis.



3. Postmultiply

$$R_Z(\gamma) = \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.120}$$

This rotates about the body $+Z$ axis, which is also the global $+X$ axis.



This time, we have arrived in the correct configuration, so we can conclude that this rotation is expressed in the current frame as

$$P_A^G = R_B^G P_A^B = R_{XYZ} P_A^B \tag{3.121}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha \\ 0 & s_\alpha & c_\alpha \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} P_A^B \tag{3.122}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} P_A^B \tag{3.123}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} P_A^B. \tag{3.124}$$

□

3.5 Rigid Motions and Homogeneous Transforms

Throughout Section 3.3 and Section 3.4 we have discussed only pure rotations; that is, a rotation without any translation. But as you may recall from Section 3.1, a *rigid motion* can result from a rotation, a translation, or both. To simplify our calculations, it is desirable to have a single representation for a rigid motion rather than a separate representation for a rotation and a translation. We discuss this single representation in this section.

To begin, we will express a rigid motion using the tools we have so far. Let R_B^G be the rotation matrix that describes the orientation of a body frame $\{B\}$ with respect to the global frame $\{G\}$, and let P_B^G be the vector from the origin of $\{G\}$ to the origin of $\{B\}$ (recall that P_B^G is shorthand for $P_{O_B}^G$, where O_B is the origin of $\{B\}$). Let P_A^B be the position vector denoting the position of point A, which is rigidly affixed to body B, in frame $\{B\}$. Then the coordinates of A with respect to $\{G\}$ are given by

$$P_A^G = R_B^G P_A^B + P_B^G. \tag{3.125}$$

Now suppose that we have three frames $\{0\}$, $\{1\}$, and $\{2\}$. To express a point A fixed in frame $\{2\}$ with respect to frame $\{0\}$, we can use (3.125) to write the two equations

$$P_A^1 = R_2^1 P_A^2 + P_2^1, \tag{3.126}$$

$$P_A^0 = R_1^0 P_A^1 + P_1^0, \tag{3.127}$$

and then we substitute the expression for P_A^1 from (3.126) into (3.127) to get

$$\begin{aligned} P_A^0 &= R_1^0(R_2^1 P_A^2 + P_2^1) + P_1^0 && (3.128)\\ &= R_1^0 R_2^1 P_A^2 + R_1^0 P_2^1 + P_1^0. && (3.129) \end{aligned}$$

Since we also could have written the relationship between P_A^0 and P_A^2 as

$$P_A^0 = R_2^0 P_A^2 + P_2^0, \tag{3.130}$$

it follows that we have the relationships

$$R_2^0 = R_1^0 R_2^1, \tag{3.131}$$

$$P_2^0 = R_1^0 P_2^1 + P_1^0. \tag{3.132}$$

Already we can see that if we have an articulated manipulator arm with many joints, expressing the rigid motion of the end-effector relative to the fixed frame $\{0\}$ will require a long chain of calculations both for the angular displacement and again for the linear displacement, using the same rotation matrices multiple times. What we would like to do instead is reduce the computation entirely to matrix multiplications. We are able to do just this with a matrix representation of a rigid motion called the **homogeneous transformation**.

In three dimensions, the homogeneous transformation is a 4×4 matrix with the following form:

$$H = \begin{bmatrix} R & P \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.133}$$

where $R \in \mathrm{SO}(3)$, $P \in \mathbb{R}^3$, and $\mathbf{0}$ is a zero row vector.

To provide some intuition about why the homogeneous transformation matrix has this form, we will briefly describe the *homogeneous coordinate system* from which the homogeneous transformation derives its name. We can take any point in a Cartesian coordinate system (x, y, z) and describe it in homogeneous coordinates as (xk, yk, zk, k), where k is some scalar. To convert from homogeneous coordinates to Cartesian coordinates, we compute $(x/k, y/k, z/k)$. Two cases are of interest here:

- If $k = 1$, then the homogeneous coordinates are called *normalized* since we can read the Cartesian coordinates directly from the homogeneous ones. (Note that in this context, normalization does not mean the same thing as normalizing a vector. It only means that $k = 1$.)

- If $k = 0$, then we cannot express the homogeneous coordinates as Cartesian coordinates because they represent a point at infinity. An easier way to think about this is that they represent a *direction*. For example, the homogeneous coordinates $(1, 0, 0, 0)$ represent the direction of the positive x-axis.

If we look back at (3.133) and think about it in terms of its column vectors, the choice of $\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$ as the final row vector starts to make sense. The first three columns are the axes of a frame, not points, so we set $k = 0$ accordingly. On the other hand, the final column is a position vector, so we set $k = 1$, which allows us to express the position vector in homogeneous coordinates without changing the x-, y-, and z-components of the vector.

Now let us return to the two rigid motions described by (3.126) and (3.127). If we represent these rigid motions as homogeneous transformations and multiply them together, the result is

$$\begin{bmatrix} R_1^0 & P_1^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_2^1 & P_2^1 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} R_1^0 R_2^1 & R_1^0 P_2^1 + P_1^0 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} R_2^0 & P_2^0 \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.134}$$

Thus, we were able to compute R_2^0 and P_2^0 like we did before, but all we had to do was multiply matrices, which is much more computationally efficient.

To apply a homogeneous transformation to a position vector such as P_A^2, all we need to do is write the vector in its normalized homogeneous form, $\begin{bmatrix}P_A^2 & 1\end{bmatrix}^T$, and then carry out the multiplication, as

$$\begin{bmatrix} P_A^0 \\ 1 \end{bmatrix} = \begin{bmatrix} R_2^0 & P_2^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} P_A^2 \\ 1 \end{bmatrix}. \tag{3.135}$$

Then we can read the Cartesian coordinates directly from the result since it is in normalized homogeneous form.

Just as it was easy to compute the inverse of a rotation matrix, it is easy to compute the inverse transformation H^{-1} from a homogeneous transformation H:

$$H^{-1} = \begin{bmatrix} R^T & -R^T P \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.136}$$

Finally, it is important to note that the current and fixed frame interpretations that we derived for rotation matrices also hold for homogeneous transformations. Given a homogeneous transformation H_1^0 relating two frames, if a second rigid motion H is applied in the current frame, we postmultiply to get

$$H_2^0 = H_1^0 H, \tag{3.137}$$

whereas if H is applied in the fixed frame, we premultiply to get

$$H_2^0 = H H_1^0. \tag{3.138}$$

3.5.1 Long-Form Example

Example 3.5 Document Viewer

Concepts reviewed: *homogeneous transforms*, *expressing coordinates in a new frame*, *composing transforms*, *inverse of a transform*.

Problem: Consider the diagram in Fig. 3.17. A document camera is centered one meter above a small object of negligible size that is centered on a square table. Find H_1^0, H_2^1, H_3^1. Using only these matrices, find H_3^2, i.e. the position and orientation of the small object in the camera's coordinate frame.

Solution: For the first homogeneous transformation matrix, we note that coordinate frames $\{0\}$ and $\{1\}$ are aligned, allowing us to construct a pure translation:

$$H_1^0 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.139}$$

The remaining two matrices have non-trivial rotation and translation components. We can infer the rotation component from the diagram in two ways:

1. **By inspection.** Inspection works well only in special cases such as this, where the axes are aligned. Recall that the column vectors of a rotation matrix R_2^1 are the three coordinate frame axes of the frame $\{2\}$ expressed

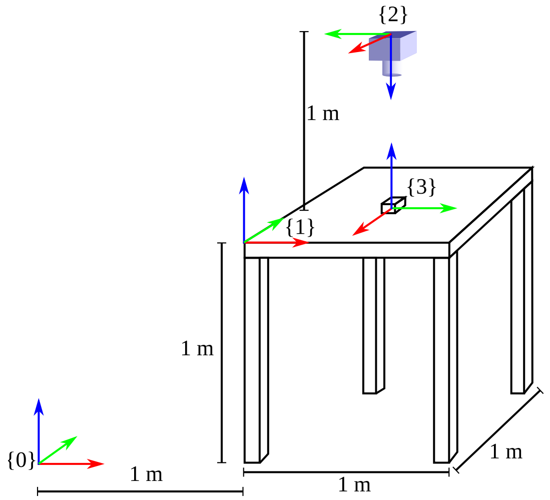


Figure 3.17: A document viewer is looking down at a small object centered on a table.

relative to the frame $\{1\}$. Then we have

$$\hat{x}_2^1 = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}^T \tag{3.140}$$

$$\hat{y}_2^1 = \begin{bmatrix} -1 & 0 & 0 \end{bmatrix}^T \tag{3.141}$$

$$\hat{z}_2^1 = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T \tag{3.142}$$

$$R_2^1 = \begin{bmatrix} \boxed{\begin{matrix} 0 \\ -1 \\ 0 \end{matrix}} & \boxed{\begin{matrix} -1 \\ 0 \\ 0 \end{matrix}} & \boxed{\begin{matrix} 0 \\ 0 \\ -1 \end{matrix}} \end{bmatrix} \tag{3.143}$$

2. **From the definition.** Using the general approach,

$$R_2^1 = \begin{bmatrix} \hat{x}_1 \cdot \hat{x}_2 & \hat{x}_1 \cdot \hat{y}_2 & \hat{x}_1 \cdot \hat{z}_2 \\ \hat{y}_1 \cdot \hat{x}_2 & \hat{y}_1 \cdot \hat{y}_2 & \hat{y}_1 \cdot \hat{z}_2 \\ \hat{z}_1 \cdot \hat{x}_2 & \hat{z}_1 \cdot \hat{y}_2 & \hat{z}_1 \cdot \hat{z}_2 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{3.144}$$

One may confuse this matrix of dot products with its transpose. Recall that $\left(R_B^G\right)^T = \left(R_B^G\right)^{-1} = R_G^B$, so one must be careful. To resolve the confusion, observe that the column that defines the $\hat{y}_2$ axis contains $\hat{y}_2$ at each element. This vector is the projection of $\hat{y}_2$ onto each of the three axes of frame $\{1\}$.

Having found the orientation of frame $\{2\}$ in frame $\{1\}$, we can now combine it with the translation component $\begin{bmatrix}\frac{1}{2} & \frac{1}{2} & 1\end{bmatrix}^T$ to get

$$H_2^1 = \begin{bmatrix} 0 & -1 & 0 & \frac{1}{2} \\ -1 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.145}$$

We follow the same procedure to find H_3^1. Using the dot product method,

$$R_3^1 = \begin{bmatrix} \hat{x}_1 \cdot \hat{x}_3 & \hat{x}_1 \cdot \hat{y}_3 & \hat{x}_1 \cdot \hat{z}_3 \\ \hat{y}_1 \cdot \hat{x}_3 & \hat{y}_1 \cdot \hat{y}_3 & \hat{y}_1 \cdot \hat{z}_3 \\ \hat{z}_1 \cdot \hat{x}_3 & \hat{z}_1 \cdot \hat{y}_3 & \hat{z}_1 \cdot \hat{z}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.146}$$

We can again construct the translation vector $\begin{bmatrix}\frac{1}{2} & \frac{1}{2} & 0\end{bmatrix}^T$ by inspection. Thus,

$$H_3^1 = \begin{bmatrix} 0 & 1 & 0 & \frac{1}{2} \\ -1 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.147}$$

Finally, we wish to construct an expression for H_3^2 using only H_1^0, H_2^1, H_3^1. One way to write H_3^2 is as a composition of matrices,

$$H_3^2 = H_1^2 H_3^1, \tag{3.148}$$

but since we do not know H_1^2, we must invert the matrix we do have, yielding

$$H_3^2 = \left(H_2^1\right)^{-1} H_3^1. \tag{3.149}$$

Recall that the definition of the inverse of a homogeneous transformation is

$$H^{-1} = \begin{bmatrix} R^T & -R^T P \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.150}$$

Since $-I_3^T = -I_3$ and $-(-I_3) = I_3$, it turns out that $\left(H_2^1\right)^{-1} = H_2^1$. Thus,

$$H_3^2 = \left(H_2^1\right)^{-1} H_3^1 = \begin{bmatrix} 0 & -1 & 0 & \frac{1}{2} \\ -1 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & \frac{1}{2} \\ -1 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.151}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.152}$$

This result confirms the result we get by inspection from Fig. 3.17.

□

3.6 Configuration Spaces

If we wish to completely describe the configuration of a robot, we must express the position of every particle that makes up the robot. We previously introduced the rigid body assumption, which allows us to define the position of every particle in a rigid body based on knowing the *pose* (position and orientation) of its coordinate frame alone. This pose could be given by a homogeneous transformation matrix, for example. The pose of a rigid body has six degrees of freedom in 3D or three degrees of freedom in 2D.

An *articulated* robot has one or more joints, which violate the rigid body assumption since joints permit internal motion within the robot. Even though the rigid body abstraction does not apply to an articulated robot as a whole, it still applies to each of the individual links. We could therefore define the pose of the whole robot of n links by using $6n$ variables (or $3n$ in 2D).

However, treating each link as a completely independent rigid body is inefficient because it neglects the fact that the links are joined together at the joints, which constrain the ways in which the links may move with respect to one another. Just as we previously observed for mobile robots, constraints produce interesting and useful motions that we can describe mathematically. The nature of these constraints and the resulting motions is the subject of Sections 3.7–3.9.

For now, we only need to know that each joint has a single degree of freedom[2] (and therefore five constraints in 3D or two constraints in 2D). We can describe the internal configuration of the robot as a simple list of the joint values for all of the joints in the robot. From these joint values, we will be able to reconstruct the full pose of each link in the *kinematic chain* of the robot.

The configuration of a robot is the vector of all the degrees of freedom required to completely specify the position of every particle. Each degree of freedom may have upper and lower bounds due to the physical mechanism having a hard stop, or it may be unconstrained. The set of possible values of the configuration vector defines the *configuration space* of a robot.

3.6.1 Spaces

A *space* is a continuous, connected set with a dimension. Since the configuration is defined by a list of variables, the configuration space is the set of all permissible configurations. The variables in the list are independent of one another, meaning that the set of permissible configurations is the combination of all possible values of each variable. We express this concept via the *Cartesian product* operator ×.

[2] More complex joints can be decomposed into multiple joints, each with one degree of freedom.

For example, we can describe the space of two real-valued variables as $\mathbb{R} \times \mathbb{R}$. We can also abbreviate $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$ and $\mathbb{R}^3 = \mathbb{R} \times \mathbb{R} \times \mathbb{R}$.

Although a real number can be used to hold the value of an angle, it is neither necessary nor desirable to do so. Because an angle wraps around, it can only take values between $-\pi$ and $+\pi$ radians. We use the symbol $\mathbb{S}$ to describe an angle, and we can similarly compose $\mathbb{S} \times \mathbb{S}$ if we have two angle variables in our configuration. However, rotations in 3D are special because consecutive rotations of the same rigid body about different axes affect one another. To describe a 3D rotation of a rigid body, we write $\mathbb{S}^3$, but this is *not* the same as $\mathbb{S} \times \mathbb{S} \times \mathbb{S}$. The latter expression describes three single rotations of different rigid bodies (such as different links in a kinematic chain). We can express an element of $\mathbb{S}^3$ by a rotation matrix, Euler angles, axis-angle, or a unit quaternion. On the other hand, to express an element of $\mathbb{S} \times \mathbb{S} \times \mathbb{S}$ we use a three-element vector. The distinction between $\mathbb{S}^3$ and $\mathbb{S} \times \mathbb{S} \times \mathbb{S}$ is further explained in Section 3.6.5.

3.6.2 External Configuration Variables

If we pick one link of the robot to be the base, then we can use its coordinate frame to be the reference frame for the whole robot. Often, this will be the link that is fixed to the ground or on wheels touching the ground. It is usually also the largest rigid body in the robot. The base link represents the pose of the robot as a whole in space, and therefore its degrees of freedom are the three or six degrees of freedom needed to express the pose of the base in 2D or 3D, respectively.

External configuration variables are the position and orientation of the robot. In the plane (such as for a ground robot), there are two degrees of translational freedom and one degree of rotational freedom. In 3D, there are three translational and three rotational degrees of freedom. An element and the set of such configuration in 2D and 3D are described as

$$\{x, y, \theta\} \in \mathbb{R}^2 \times \mathbb{S}, \tag{3.153}$$

$$\begin{bmatrix} x \\ y \\ z \\ \alpha \\ \beta \\ \gamma \end{bmatrix} \in \mathbb{R}^3 \times \mathbb{S}^3. \tag{3.154}$$

In Equation 3.154, we use Euler angles to express an element of $\mathbb{S}^3$.

3.6.3 Internal Configuration Variables

The internal degrees of freedom are defined by the angles of the joints (or positions, if they are prismatic joints). Given a robot with n revolute joints and m prismatic joints, the internal configuration is an element of a set:

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \\ d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix} \in \prod_{i=1}^{n} \mathbb{S} \times \prod_{j=i}^{m} \mathbb{R}, \tag{3.155}$$

where the notation $\Pi_{i=1}^{n} X$ means the Cartesian product of n copies of the set X.

3.6.4 Configuration Space

Suppose we have a robot in 3D with an $(n+m)$-link arm, with n revolute joints and m prismatic joints. A point in the robot's configuration space is an element of the set

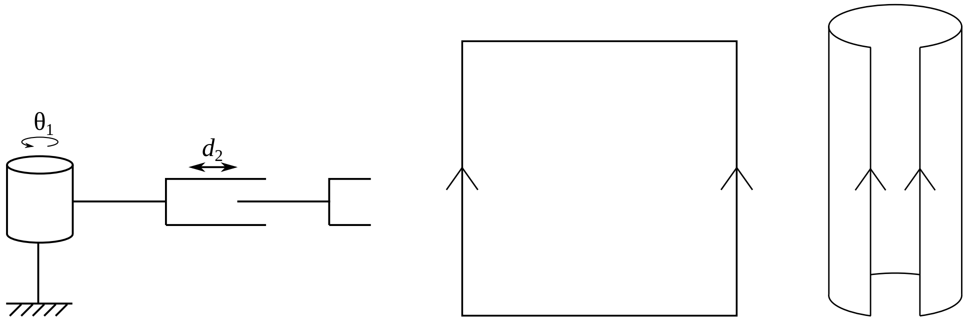
$$\mathbb{R}^3 \times \mathbb{S}^3 \times \prod_{i=1}^{n} \mathbb{S} \times \prod_{j=i}^{m} \mathbb{R} \tag{3.156}$$

that specifies a valid position for every particle in the entire articulated robot.

The configuration vector includes both internal and external degrees of freedom as applicable. For example, the KUKA youBot mobile manipulator robot has a mobile base that moves on the floor with three degrees of freedom as well as an arm with five joints for five internal degrees of freedom. Thus, the KUKA youBot has a total of eight degrees of freedom in its configuration. This model deliberately neglects the orientation of each wheel on its axle since that does not affect the performance of the robot.

3.6.5 The Topology of Configuration Spaces

Topology is the field of math that studies the connectivity of shapes under stretching but not cutting. We can use concepts from topology to build intuition about the differences in the structure of configuration spaces induced by differences in their physical mechanisms. This idea is useful to us here because of the structure of the



(a) A 2-joint manipulator with a fixed base, a revolute joint, and a prismatic joint.

(b) The gluing diagram for one revolute and one prismatic joint shows horizontal wraparound but not vertical wraparound.

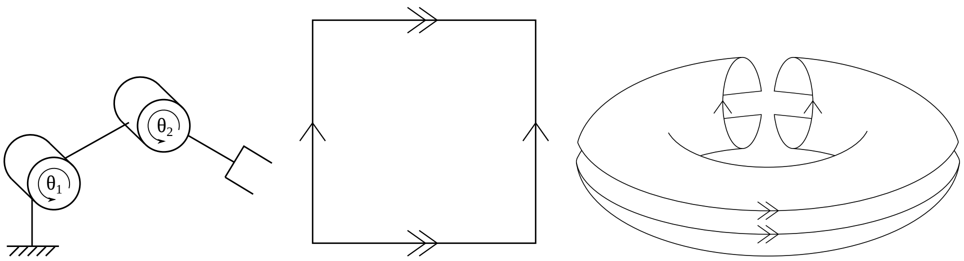
(c) Matching identified edges.

Figure 3.18: The gluing diagram for an arm with one revolute and one prismatic joint is a cylinder. The identical arrows are identified, which means that the square is folded until the edges overlap. The result is a cylinder. This shape represents the space $\mathbb{R} \times \mathbb{S}$.

set $\mathbb{S}$, which is properly rendered like a circle: you can start from the orientation 0 and move in the positive direction, and you will keep passing 0 every 2π that you move.

A *gluing diagram* is a method of describing non-flat shapes on the flat page. We can use a gluing diagram to help visualize the structure of two-joint robots in a 2D drawing. It is a square with arrows on some or all sides, annotating the way that edges are *identified* with each other. When two points are identified, it means that we regard them as the same point. Consequently, two edges with the same marking are regarded as the same edge, and the points on one side of the edge are in the neighborhood of the points on the other side. One could therefore draw a continuous path beginning on one side of the edge and continuing onto the other side without there being a discontinuity in the path.

We begin with a plain square, which represents the set $\mathbb{R}^2$. The square represents the configuration space of an arm with two prismatic joints. The x-coordinate corresponds to one joint value, and the y-coordinate corresponds to the other joint value. Thus, a point in the square represents a configuration of the robot arm. Motion of the arm can instantaneously be in any linear combination of the x- and y-axes, corresponding to any combination of motion of the two joints. The square is a representation of the plane since motion never wraps around in any direction. Now we consider what happens if one or both prismatic joints are replaced with a revolute joint.



(a) This mechanism has two independent revolute joints.

(b) The gluing diagram for the two-link arm shows that both pairs of opposing edges are identified.

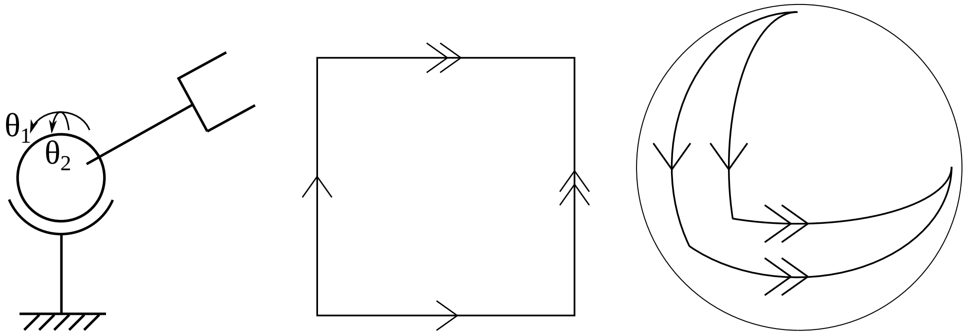
(c) The resulting figure is a torus. From any point, you can move along two distinct axes, resulting in circumnavigating two distinct holes (the donut hole and the interior of the torus).

Figure 3.19: The gluing diagram for an arm with two revolute joints is a torus. The corresponding space is $\mathbb{S} \times \mathbb{S}$.

If a two-joint arm has one revolute joint, then the square does not adequately represent its topology, since the circular motion of the revolute joint must be cut to get a flat square, and cutting is forbidden. We can restore the circular motion of the revolute joint by gluing the edges back together where they were cut (Fig. 3.18). To indicate gluing, we add arrows in pairs to the edges of the square. A pair of arrows indicate that the corresponding edges are *identified*, meaning they are the same edge for the purpose of folding the square into 3D. There is a smooth, one-to-one correspondence between every point on one edge and every point on the other edge. After assigning arrows to edges, the gluing diagram gets folded into a shape in 3D so that the arrows all match up. When doing this, remember that topology permits stretching and bending a shape. After folding into 3D, the two-joint arm with one revolute and one prismatic joint results in a cylinder.

Arrows in the gluing diagram are assigned according to the physical mechanism, and the assignments of those arrows in turn determine the form of the 3D shape. Recalling that revolute joints cause edges to be identified, whereas prismatic joints do not, we can see how different shapes emerge. For an arm with one revolute and one prismatic joint, only one pair of opposite edges are identified, and the result is a cylinder (Fig. 3.18). For an arm with two links and two revolute joints, both pairs of opposite edges are identified, and the result is a torus (Fig. 3.19). For an arm with a spherical joint — that is, two one-degree-of-freedom revolute joints whose axes meet at a point — the gluing diagram shows the identified edges adjacent, and the resulting shape is a sphere (Fig. 3.20).

The mechanisms differ in the number and type of *holes* present in the 3D



(a) This spherical joint rotates in two axes at a single point. We write it as two distinct revolute joints.

(b) These two revolute joints are coupled. We place the identified edges adjacent instead of opposite like in Fig. 3.19b.

(c) Identifying the adjacent edges causes the gluing diagram to fold differently from the torus such that motion from a point in any direction will circumnavigate around the same hole (the sphere's interior).

Figure 3.20: The gluing diagram for a spherical joint with two degrees of freedom is a sphere. The corresponding space is $\mathbb{S}^2$.

shape resulting from their gluing diagrams. These holes tell us something about the mechanism. For example, in a spherical joint, the rotations are coupled in the manner we have seen for 3D rotations: a rotation in one axis affects the amount of rotation in the other axis. Consequently, the shape that results is a sphere, which surrounds a single hole. In contrast, when we have two distinct revolute joints with parallel axes, the resulting shape is a torus, which has two independent holes. One hole is the donut hole we are used to, and the second hole is the interior of the donut where the dough would be. Moving around one hole does not affect the position with respect to the other hole, which tells us that these two joints are uncoupled.

Gluing diagrams serve to give us intuition, but they only work for two-joint mechanisms. However, the intuition we have gained about numbers of holes and the corresponding coupling of joint values extends into arbitrary dimensional configuration spaces. In addition, we would like to add some geometry to these concepts, so that we can measure the positions of links and especially the end-effector of the arm. In the following section, we formalize the case of arbitrary numbers of joints in an arm, and we study how to parameterize the corresponding links.

3.7 Forward Kinematics for Manipulator Arms

Now that we have all the tools we need to represent rigid displacements, we turn to the first of two crucial problems in kinematics which we will study in the context of manipulator arms. In this section, we will examine the **forward kinematics** problem, in which we need to determine the position and orientation of the end-effector of an arm given the configuration of its joints. If we were to solve this problem for our own human arms, we would ask, "Given the current configuration of my shoulder, elbow, and wrist, what is the position and orientation of my hand?"

In these notes, we will only consider joints with one degree of freedom (like the human elbow). Recall from Section 1.1.2 that there are two basic forms of joints with one degree of freedom: *revolute joints*, which allow rotation about a single axis, and *prismatic joints*, which allow linear motion along a single axis. It is possible to construct more complex joints with more degrees of freedom (like the human shoulder and wrist) from these basic joints, so our choice to only consider joints with one degree of freedom does not give rise to any loss of generality.

Since a joint connects two links, a manipulator arm with n joints will have $n+1$ links. We will adopt the convention of numbering the joints from 1 to n and the links from 0 to n, starting from the base of the arm. As a result, joint i connects link $i-1$ to link i, and when joint i is actuated, link i moves. Note that since there is no joint 0, link 0 cannot be actuated; it is fixed to the base of the arm.

Given that the joints we are considering possess one degree of freedom, we only need one variable per joint to describe the configuration. We therefore associate a single joint variable q_i with each joint i. What the joint variable represents depends on the type of joint in question. If joint i is revolute, then q_i is the angle of displacement, which we call θ_i. Otherwise, if joint i is prismatic, then q_i is the linear displacement, which we call d_i.

Since our goal is to determine the position and orientation of the end-effector, we need to attach a frame to each link, which we do by attaching frame $\{i\}$ to link $\{i\}$. Having done that, we can consider the homogeneous transformation matrices that relate the links to each other. These matrices are not constant because as the links move, the attached frames move with them. Rather, each matrix is a function of a single joint variable. If we let H_i be the homogeneous transform that gives the position and orientation of frame $\{i\}$ relative to frame $\{i-1\}$, then H_i is a function of q_i. We can express this concisely as

$$H_i = H_i(q_i). \tag{3.157}$$

The homogeneous transform that allows us to solve the forward kinematics problem is H_n^0, which gives the position and orientation of $\{n\}$ (the frame of the end-effector) relative to frame $\{0\}$ (the frame of the link fixed to the base). H_n^0 is a function of *all* the joint variables in the manipulator arm. In other words,

$$H_n^0 = H_n^0(q_1, q_2, \ldots, q_n). \tag{3.158}$$

As we saw in Section 3.5, we can compose homogeneous transforms to relate different frames to each other. Knowing this, we can compute H_n^0 as

$$H_n^0 = H_1 H_2 \cdots H_n, \tag{3.159}$$

where each homogeneous transform H_i has the form

$$H_i = \begin{bmatrix} R_i^{i-1} & P_i^{i-1} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.160}$$

and the homogeneous transform H_n^0 has the form

$$H_n^0 = \begin{bmatrix} R_n^0 & P_n^0 \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.161}$$

Of course, this is not the end of the story; we still need to figure out how to determine each homogeneous transform H_i that is required to do the computation. It is important to realize that there is not one correct set of homogeneous transforms. After all, all we said with regard to frame placement is that we attach frame $\{i\}$ to link $\{i\}$. But where on the link do we attach it? At what orientation do we attach it? The way in which we attach the frames completely changes which homogeneous transforms are correct for a given manipulator arm.

Rather than attaching the frames in whatever way we choose and leaving other roboticists to guess how we did it, we will instead attach the frames according to the *Denavit-Hartenberg convention*. The purpose of this convention is to standardize how the frames are attached and streamline the process of attaching them. As a word of caution, although the Denavit-Hartenberg convention is intended to standardize, some authors use slightly different versions of the convention. Thus, depending on which version is used, the frames will be attached slightly differently. We will only use one version of the convention in these notes.

3.7.1 The Denavit-Hartenberg Convention

As previously stated, attaching frames in accordance with the Denavit-Hartenberg convention is wise because it is a standard among roboticists. It turns out that there is another important benefit to using this convention that will simplify our kinematic analysis. Consider the fact that an arbitrary homogeneous transform

$H_i(q_i)$ needs six parameters to characterize it: three for the rotation component, and three for the translation component. Since $H_i(q_i)$ is a function of a single variable, five of these parameters are constant whereas the sixth is variable.

The power of using the the Denavit-Hartenberg convention is that we only need *four* parameters, not six, to derive the homogeneous transform for a given link. The reason we are able to reduce the number of parameters required is that the Denavit-Hartenberg convention restricts how we attach a frame to a link. Intuitively, if the origin of a frame is always going to be attached at the same place on a link, and the axes are always going to be oriented in a certain manner, then we should need fewer parameters than we would if we had complete freedom in those choices.

Before we describe how the frames need to be assigned, we will first discuss the four parameters that we need to derive the homogeneous transforms for the arm. These parameters are referred to as the *Denavit-Hartenberg parameters* and are often called DH parameters for short. To define them, we must first define a *joint axis*, which for a revolute joint is the axis of rotation and for a prismatic joint is the direction of motion, and the *common normal* of a link, which is the shortest line segment orthogonal to the two joint axes that the link connects. Now we can define the DH parameters as follows (see Fig. 3.21 for a visualization of each one):

1. θ_i, the *joint angle*, is the angle between the common normals of links $i-1$ and i.

2. d_i, the *link offset*, is the distance between where the common normals of links $i-1$ and i intersect the axis of joint i,

3. a_i, the *link length*, is the length of the common normal of link i,

4. α_i, the *link twist*, is the angle between the axes of joints i and $i+1$ measured about link i's common normal[3],

The next step is to attach a frame to each link according to the Denavit-Hartenberg convention. There are two requirements for how we attach the frames:

1. The axis $\hat{x}_i$ must be perpendicular to the axis $\hat{z}_{i-1}$.

2. The axis $\hat{x}_i$ must intersect the axis $\hat{z}_{i-1}$.

As we mentioned previously, there are slightly different versions of the Denavit-Hartenberg convention; this is because there are multiple ways to attach frames

[3] To help envision how this angle is measured, imagine that you are holding a pair of nunchucks with the sticks parallel and the chain taut and perpendicular to the sticks. Now imagine twisting one of the sticks such that the chain is the axis of rotation and the stick you are twisting remains perpendicular to the chain. The amount you twisted the stick is the link twist.

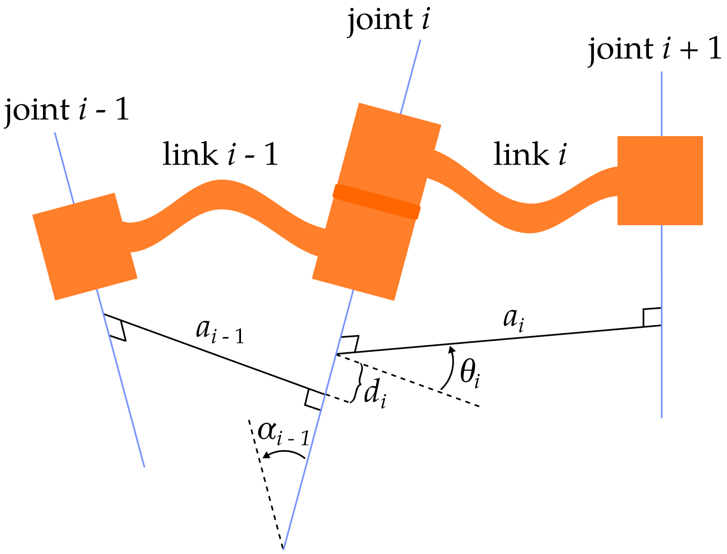


Figure 3.21: Denavit-Hartenberg parameters. The joint axes are shown in blue, and the common normals are the line segments perpendicular to the joint axes.

while satisfying the two constraints given above. In these notes we will use the following procedure to attach the frames:

1. For axes $\hat{z}_0, \ldots, \hat{z}_{n-1}$, we assign $\hat{z}_i$ to be the axis of actuation for joint $i+1$.

2. Next, we attach the base frame $\{0\}$ by selecting any point on $\hat{z}_0$ to be its origin and then choosing $\hat{x}_0$ and $\hat{y}_0$ in accordance with the right-hand rule.

3. Now we use the following iterative procedure to finish setting up frames $\{1\}$ through $\{n-1\}$, in which we use frame $\{i-1\}$ to establish frame $\{i\}$:

   (a) If axes $\hat{z}_i$ and $\hat{z}_{i-1}$ are *not* coplanar, attach the origin of frame $\{i\}$ to the point where the common normal of link i intersects $\hat{z}_i$. Conversely, if axes $\hat{z}_i$ and $\hat{z}_{i-1}$ *are* coplanar, then we have two cases. If $\hat{z}_i$ and $\hat{z}_{i-1}$ intersect, we attach the origin of frame $\{i\}$ at the point of intersection. Otherwise, if $\hat{z}_i$ and $\hat{z}_{i-1}$ are parallel, then we attach the origin of frame $\{i\}$ at the point along $\hat{z}_i$ such that $d_i = 0$.

   (b) Assign $\hat{x}_i$ to be the common normal through the origin of frame $\{i\}$ and pointing away from $\hat{z}_{i-1}$. In the case where $\hat{z}_i$ and $\hat{z}_{i-1}$ intersect, assign $\hat{x}_i$ to be normal to the plane formed by $\hat{z}_i$ and $\hat{z}_{i-1}$. In this case, there is an arbitrary choice of the direction of $\hat{x}_i$ between two directions both orthogonal to that plane.

(c) Assign $\hat{y}_i$ in accordance with the right-hand rule.

4. Finally, we need to attach frame $\{n\}$, which is the frame of the end-effector. Typically we choose the origin of $\{n\}$ to be a point symmetric between the fingers of the gripper, but we might have reason to choose a different point (for example, we might do this if the manipulator arm can swap out its end-effector for a different one). There are also multiple ways we might choose the axes for the frame, but we will describe the approach for a standard two-finger gripper. We choose $\hat{z}_n$ to be the *approach* axis, which is the direction that the gripper moves when preparing to grasp an object. We choose $\hat{y}_n$ to be the *sliding* axis, which is the axis along which the fingers of the gripper slide when opening and closing. Lastly, we choose $\hat{x}_n$ to be the *normal* axis, which is the direction normal to the plane formed by $\hat{z}_n$ and $\hat{y}_n$.

Now that we have finished attaching frames, we can derive each homogeneous transform H_i by composing the following four steps:

1. Rotate about the current $\hat{z}$-axis ($\hat{z}_{i-1}$) by θ_i radians.
2. Translate along the current $\hat{z}$-axis ($\hat{z}_{i-1}$) by d_i units.
3. Translate along the current $\hat{x}$-axis ($\hat{x}_i$) by a_i units.
4. Rotate about the current $\hat{x}$-axis ($\hat{x}_i$) by α_i radians.

Note that since each transform is applied in the current frame, we are technically applying steps 2–4 in intermediate frames that do not have names. However, the intermediate frames' $\hat{z}$-axes coincide with $\hat{z}_{i-1}$, and after step 2, the remaining frames' $\hat{x}$-axes coincide with $\hat{x}_i$. This is why we do not need to write the names of the intermediate frames explicitly.

Applying the four steps in order is equivalent to composing the transforms as

$$\begin{aligned} H_i &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{3.162}$$

And with that, we have everything we need to compute the forward kinematics of a manipulator arm. Although it took some work to get here, once we have all the homogeneous transforms we need, all we need to do is plug in joint variables and multiply matrices. In the next section, we will see that the inverse problem is not nearly as straightforward.

3.8 Inverse Kinematics for Manipulator Arms

In the previous section, we discussed how to solve for the position and orientation of a manipulator's end-effector given the values of its joint variables. We will now consider the inverse of this problem, which is to find values of the joint variables that place the end-effector at a desired position and orientation in space. In general, this problem of **inverse kinematics** is more difficult than forward kinematics since the equations we must solve are often complicated and nonlinear, and there may exist many solutions or no solutions.

3.8.1 Solution Existence and Workspace

In order for at least one solution to exist, the desired goal point must lie within the manipulator's **(reachable) workspace**, which is the volume of space that the robot can reach in at least one orientation. Depending on the manipulator, there may also be a subset of this space called the **dexterous workspace**, which is all the points that the robot can reach from any arbitrary orientation of its end-effector.

Consider the manipulator arm depicted in Fig. 3.22. The manipulator has two revolute joints and three links (note that the fixed link, link 0, has length zero). Assuming that the manipulator can rotate its joints a full 360 degrees, the reachable workspace of this manipulator is a disk where the outer radius is the sum of its two nonzero link lengths and the inner radius is the difference between its two nonzero link lengths. Does this manipulator have any dexterous workspace? As shown in Fig. 3.22a, the manipulator can reach a point on the boundary of its workspace from only one orientation, so the boundaries of the disk cannot constitute dexterous workspace. Fig. 3.22b shows that a point inside of the manipulator's workspace can be reached from exactly two orientations. Thus, no point in this manipulator's workspace can be reached from any arbitrary orientation of the end-effector, so the manipulator has no dexterous workspace. Note, however, that if the two nonzero links were the same length, the workspace would be a circle, and there would be a single point of dexterous workspace: the origin of the circle.

3.8.2 Multiple Solutions

Although the manipulator shown in Fig. 3.22 can reach points on the interior of its workspace from two orientations, there is no point it can reach from multiple joint configurations with the same position *and* orientation of its end-effector. Adding another joint and link would allow for multiple solutions of this type (see Fig. 3.23).

The number of solutions to an inverse kinematics problem is a function of the manipulator's DH parameters and the range of motion of each joint. In general, the more nonzero DH parameters, the more solutions there will be. While having multiple solutions (and thus multiple ways to reach a goal) may be beneficial to the task at hand, it may also be problematic since the system must be able to choose one solution. The selection criteria vary, but reasonable factors to take into account include potential obstacle collision and how far each joint will need to move.

3.8.3 Solution Methods

Whereas we can always solve the forward kinematics problem by using our straightforward algorithm of multiplying homogeneous transformation matrices, there are no general algorithms we can use for the inverse kinematics problem that guarantee a solution if one exists. The strategies for solving the inverse kinematics problem can be divided into **closed-form solutions**, which are exact and return all solutions, and **numerical solutions**, which are approximate and usually return just one solution. Closed-form solutions are typically preferred since they are often much faster than iterative numerical searches, and they also ease the process of developing of rules to choose among multiple solutions. We therefore focus our attention on two classes of closed-form solutions: **algebraic solutions** and **geometric solutions**.

Example 3.6 Solving the inverse kinematics problem algebraically

Consider the planar manipulator in Fig. 3.24 and its associated DH parameters. We can use our method for solving the forward kinematics problem to obtain the homogeneous transforms

$$H_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; H_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

which we then multiply together to get the kinematic equations

$$H_1^0 H_2^1 = H_2^0 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1c_1 + a_2c_{12} \\ s_{12} & c_{12} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.163}$$

where $c_1 = \cos(\theta_1)$, $s_{12} = \sin(\theta_1 + \theta_2)$, and so forth.

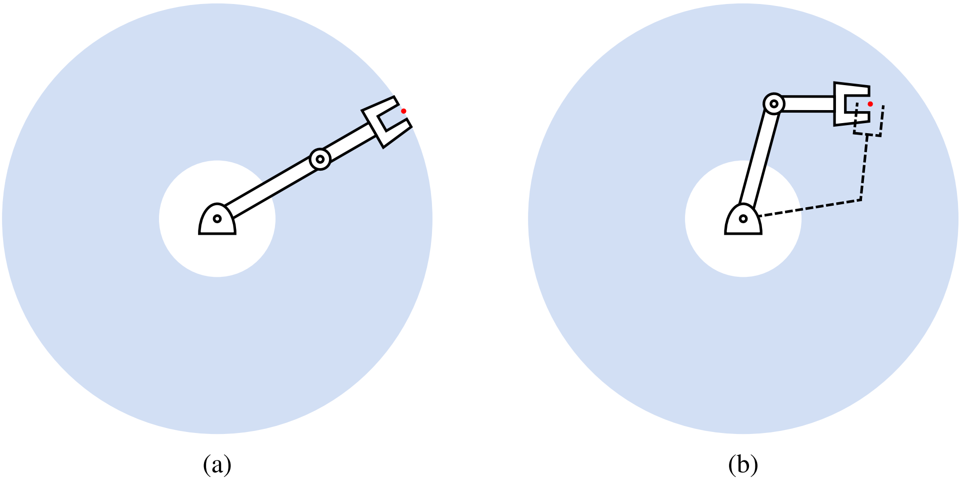


Figure 3.22: On the boundary of its reachable workspace, this manipulator can reach a point from only one orientation, as shown in (a). Inside of its reachable workspace, two orientations per point are possible: one with the manipulator's elbow up, the other with its elbow down, as shown in (b). The "elbow down" orientation is given in dashed lines.

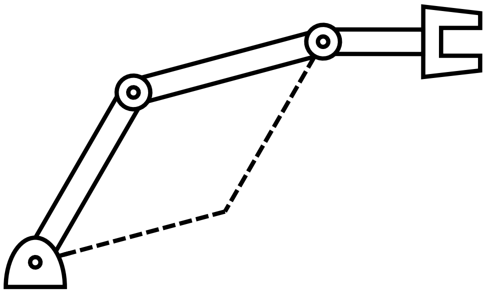


Figure 3.23: A manipulator with three revolute joints that has multiple joint configurations corresponding to the same position and orientation of its end-effector. An alternate solution is shown in dashed lines.

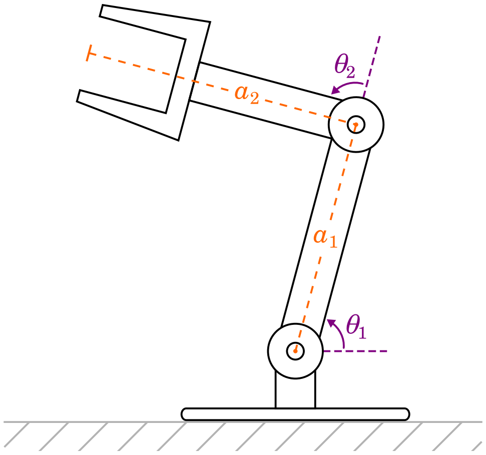


Figure 3.24: A planar manipulator with two revolute joints. The nonzero DH parameters are labeled.

Since the manipulator is planar, goal points for the end-effector can most easily be specified with an (x, y)-position and an orientation angle ϕ. Thus, all attainable goals must lie within the subspace implied by the homogeneous transformation

$$H_2^0 = \begin{bmatrix} c_\phi & -s_\phi & 0 & x \\ s_\phi & c_\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.164}$$

Equating (3.163) and (3.164) gives us the nonlinear equations which must be solved for the joint parameters θ_1 and θ_2:

$$c_\phi = c_{12}, \tag{3.165}$$
$$s_\phi = s_{12}, \tag{3.166}$$
$$x = a_1 c_1 + a_2 c_{12}, \tag{3.167}$$
$$y = a_1 s_1 + a_2 s_{12}. \tag{3.168}$$

To solve these equations, we first square (3.167) and (3.168) and add them to get

$$\begin{aligned}
x^2 + y^2 &= (a_1 c_1 + a_2 c_{12})^2 + (a_1 s_1 + a_2 s_{12})^2 \\
&= a_1^2(c_1^2 + s_2^2) + a_2^2(c_{12}^2 + s_{12}^2) + 2a_1 a_2(c_1 c_{12} + s_1 s_{12}) \\
&= a_1^2(1) + a_2^2(1) + 2a_1 a_2(c_1(c_1 c_2 - s_1 s_2) + s_1(c_1 s_2 + s_1 c_2)) \\
&= a_1^2 + a_2^2 + 2a_1 a_2(c_1^2 c_2 + s_1^2 c_2) \\
&= a_1^2 + a_2^2 + 2a_1 a_2(c_2(c_1^2 + s_1^2)) \\
&= a_1^2 + a_2^2 + 2a_1 a_2(c_2(1)) \\
&= a_1^2 + a_2^2 + 2a_1 a_2 c_2 \qquad (3.169)
\end{aligned}$$

using the trigonometric Pythagorean identity and angle sum identities.

Next, we solve (3.169) for c_2, which yields

$$c_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1 a_2}. \qquad (3.170)$$

The right-hand side of this equation must have a value between -1 and 1 in order for a solution to exist; otherwise, the goal point is out of the manipulator's reach. Assuming that the goal falls within the workspace, we use the Pythagorean identity to write an expression for s_2,

$$s_2 = \pm\sqrt{1 - c_2^2}, \qquad (3.171)$$

where the choice of sign corresponds to the elbow-up and elbow-down solutions. Lastly, we use the two-argument arctangent to write an expression for θ_2,

$$\theta_2 = \text{atan2}(s_2, c_2). \qquad (3.172)$$

Now that we have obtained θ_2, we can solve (3.167) and (3.168) for θ_1. Since c_1 and c_2 are the only unknowns, we rewrite the equations to isolate them, as

$$\begin{aligned}
x &= a_1 c_1 + a_2(c_1 c_2 - s_1 s_2) \\
&= c_1(a_1 + a_2 c_2) - s_1(a_2 s_2) \\
&= k_1 c_1 - k_2 s_1, \qquad (3.173)
\end{aligned}$$

$$\begin{aligned}
y &= a_1 s_1 + a_2(c_1 s_2 + s_1 c_2) \\
&= s_1(a_1 + a_2 c_2) + c_1(a_2 s_2) \\
&= k_1 s_1 + k_2 c_1, \qquad (3.174)
\end{aligned}$$

where

$$\begin{aligned} k_1 &= a_1 + a_2 c_2, \\ k_2 &= a_2 s_2, \end{aligned} \tag{3.175}$$

are constants. If we then define

$$r = \sqrt{k_1^2 + k_2^2} \tag{3.176}$$

and

$$\gamma = \text{atan2}(k_2, k_1), \tag{3.177}$$

we can rewrite the constants as

$$\begin{aligned} k_1 &= r \cos \gamma, \\ k_2 &= r \sin \gamma, \end{aligned} \tag{3.178}$$

which allows us to write (3.173) and (3.174) as

$$\frac{x}{r} = \cos\gamma \cos\theta_1 - \sin\gamma \sin\theta_1, \tag{3.179}$$

$$\frac{y}{r} = \cos\gamma \sin\theta_1 + \sin\gamma \cos\theta_1, \tag{3.180}$$

or, using the angle sum identities,

$$\cos(\gamma + \theta_1) = \frac{x}{r}, \tag{3.181}$$

$$\sin(\gamma + \theta_1) = \frac{y}{r}. \tag{3.182}$$

Finally, we use the two-argument arctangent to write

$$\gamma + \theta_1 = \text{atan2}\left(\frac{y}{r}, \frac{x}{r}\right) = \text{atan2}(y, x), \tag{3.183}$$

and then subtract γ from both sides to get

$$\theta_1 = \text{atan2}(y, x) - \text{atan2}(k_2, k_1). \tag{3.184}$$

If x and y are both zero, then (3.184) is undefined, which makes θ_1 arbitrary. Whenever θ_1 is defined, its sign is determined by the choice of θ_2 above since the sign of θ_2 impacts the sign of k_2, therefore affecting the sign of θ_1.

Example 3.7 Solving the inverse kinematics problem geometrically

When we use a geometric approach to solve the inverse kinematics problem, we decompose the spatial geometry of a manipulator arm into several geometric problems in the plane. Fig. 3.25 shows the geometry of a planar manipulator with the position of the wrist given as (x, y). Using the rules of planar geometry, we will solve for the joint variables θ_1 and θ_2 given this wrist position. Note that this problem is different than our typical inverse kinematics problem, which is to solve for all of the joint variables given the position and orientation of the end-effector, but it is still an inverse kinematics problem just the same.[4]

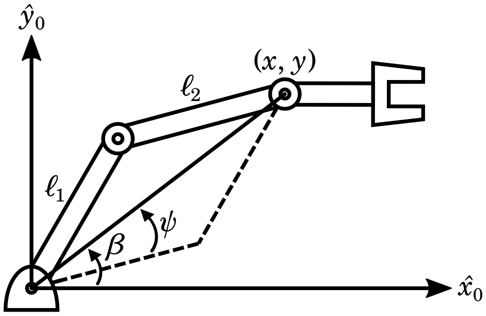


Figure 3.25: The geometry of a planar manipulator with wrist position (x, y). An alternate solution is shown in dashed lines.

To begin, observe that we know all three side lengths of the triangle formed by the solid line and the arm: ℓ_1, ℓ_2, and $\sqrt{x^2 + y^2}$ by the Pythagorean theorem. Planar geometry tells us that in order for this triangle to exist, $\sqrt{x^2 + y^2}$ must be less than or equal to $\ell_1 + \ell_2$, the sum of the lengths of link 1 and link 2. If this constraint is not met, then the goal position for the wrist, (x, y), is not within the manipulator's workspace, and there are no solutions.

Assuming that the goal position is reachable, we use the law of cosines to determine the angle opposite the solid diagonal and solve for θ_2. Recall that joint angle θ_i is defined as the angle between the common normals of links $i - 1$ and i; thus, θ_2 is the angle between the common normals of links 1 and 2. If θ_2 were 0, then the angle formed by links 1 and 2 would be 180° (see Fig. 3.26a). Hence, in general, the angle formed by links 1 and 2 is $180^\circ + \theta_2$, as shown in Fig. 3.26b.

The law of cosines tells us that

$$x^2 + y^2 = \ell_1^2 + \ell_2^2 - 2\ell_1\ell_2 \cos(180^\circ + \theta_2), \tag{3.185}$$

[4] There are practical reason to solve inverse kinematics problems of this type. Once such reason is bracing. When writing by hand, a human will typically rest the wrist of her writing hand on a table to eliminate noise from proximal muscles (and therefore have better control of the writing implement). We may choose to do something similar with an articulated arm for more precise control.

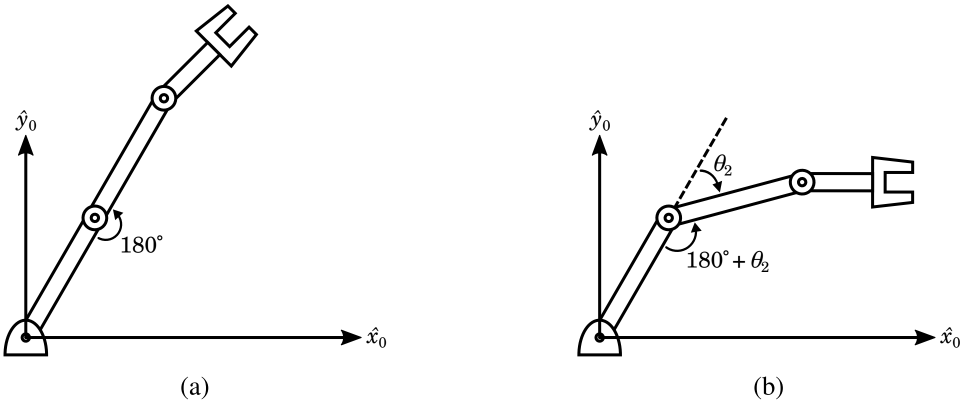


Figure 3.26: If $\theta_2 = 0$, then the angle formed by links 1 and 2 is $180°$, as shown in (a). In general, the angle is $180° + \theta_2$, as shown in (b). Note that the θ_2 shown in (b) is negative.

which we may rewrite as

$$\cos(\theta_2) = \frac{x^2 + y^2 - \ell_1^2 - \ell_2^2}{2\ell_1\ell_2} \tag{3.186}$$

using the fact that $\cos(180° + \theta_2) = -\cos(\theta_2)$. We solve (3.186) only for values of θ_2 between 0 and $-180°$ since the triangle would not exist otherwise. Note that we may find the alternate solution of Fig. 3.25 by multiplying θ_2 by -1.

Now we solve for θ_1 by finding expressions for the angles β and ψ shown in Fig. 3.25. Since β could be in any quadrant (depending on the signs of x and y), we must use the two-argument arctangent:

$$\beta = \text{atan2}(y, x). \tag{3.187}$$

Another application of the law of cosines gives us an expression for ψ,

$$\cos\psi = \frac{x^2 + y^2 + \ell_1^2 - \ell_2^2}{2\ell_1\sqrt{x^2 + y^2}}, \tag{3.188}$$

which we solve only for values of ψ between 0 and $180°$ to preserve the given geometry. Then we have

$$\theta_1 = \begin{cases} \beta + \psi & \text{if } \theta_2 < 0, \\ \beta - \psi & \text{if } \theta_2 > 0, \\ \beta & \text{otherwise.} \end{cases} \tag{3.189}$$

Thus, we have solved for all required joint variables. The remaining joint variable, θ_3, is independent of the wrist position and therefore can be any valid value.

3.9 Velocity Kinematics

In Sections 3.7 and 3.8, we discussed how to relate the position and orientation of a manipulator's end-effector to the values of its joint variables. We are still missing a key piece of the puzzle, however. The frames we have studied thus far have all been static, and when we have talked about moving a point, it was always an instantaneous rigid body displacement. In this section, we introduce *continuous motion* of one frame with respect to another. We begin by defining linear and angular velocity and show how to study the velocity of a rigid body as the motion of coordinate frames relative to each other. We then investigate the relationship of the velocity of a manipulator's end-effector to the velocities of its joints.

3.9.1 Linear and Angular Velocity

Recall that the position of a particle A with respect to a frame $\{B\}$ is given by the position vector P_A^B. We define the derivative of this vector relative to $\{B\}$ as

$$V_{P_A^B}^B = \frac{d}{dt} P_A^B = \lim_{\Delta t \to 0} \frac{P_A^B(t + \Delta t) - P_A^B(t)}{\Delta t}, \tag{3.190}$$

which can be thought of as the *linear velocity* of A. It is important to specify the frame in which the differentiation is done since the position vector may vary in time differently with respect to different frames. For example, if A is fixed to $\{B\}$, then $V_{P_A^B}^B$ will be zero since P_A^B never varies in time with respect to $\{B\}$.

Whereas linear velocity is the property of a single point, *angular velocity* is a property of the attached coordinate frame. The angular velocity of a frame $\{B\}$ rotating with respect to frame $\{G\}$ is denoted as

$$\Omega_B^G = \mathbf{u}\frac{d\theta}{dt}. \tag{3.191}$$

Here, $\mathbf{u}$ is a unit vector in the direction of the axis of rotation, and θ is the angle between $\mathbf{u}$ and a perpendicular from any point of the body. Thus, at any moment in time, the direction of Ω_B^G gives the instantaneous axis of rotation of $\{B\}$ relative to $\{G\}$, and the magnitude of Ω_B^G gives the instantaneous speed of rotation.

3.9.2 Linear and Angular Velocity of Rigid Bodies

Through the analysis of moving coordinate frames, we will use our definitions from above to study the motion of a rigid body induced by its velocity. We will first look at linear velocity and angular velocity independently, and then we will combine them to describe the simultaneous linear and angular velocity of a rigid body.

Linear Velocity

Consider a frame $\{B\}$ attached at point O_B to a rigid body. The position of a point A is given by the vector P_A^B. We would like to describe the motion of A with respect to the global-fixed frame $\{G\}$ (see Fig. 3.27).

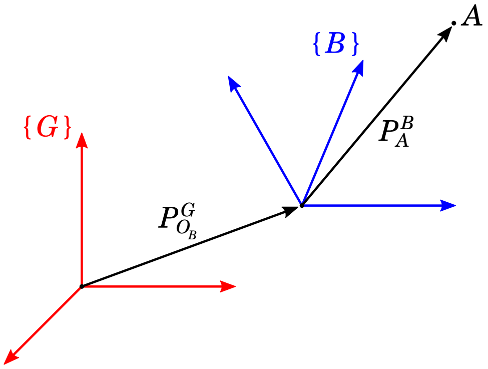


Figure 3.27: Frame $\{B\}$ is translating with respect to to frame $\{G\}$.

If we assume the relative orientation of $\{G\}$ and $\{B\}$ remains constant, then all we need to do is take the derivatives of the two vectors in Fig. 3.27 with respect to $\{G\}$ and sum them. We carry out the differentiation as follows:

1. The first vector is $P_{O_B}^G$, which gives the position of the origin of $\{B\}$ relative to $\{G\}$. Its derivative is therefore $V_{P_{O_B}^G}^G$, but to reduce the number of subscripts we must write, we will use the simplified notation V_B^G to describe the velocity of the origin of frame $\{B\}$ relative to frame $\{G\}$.

2. The second vector is P_A^B. Since this vector is expressed relative to $\{B\}$, we need to include the rotation matrix that accomplishes the change to frame $\{G\}$, yielding the derivative $R_B^G V_{P_A^B}^B$.

Hence, the linear velocity of A is

$$V_{P_A^B}^G = V_B^G + R_B^G V_{P_A^B}^B, \tag{3.192}$$

under the assumption that R_B^G does not change with time.

Angular Velocity

Suppose we have a body-fixed frame $\{B\}$ with origin coincident with the origin of the global-fixed frame $\{G\}$ and with zero linear velocity (so that its origin always stays coincident). Like before, the position of a point A is given by the vector P_A^B. As Fig. 3.28 shows, $\{B\}$ is rotating relative to $\{G\}$ with angular velocity Ω_B^G. We wish to determine how point A moves with respect to $\{G\}$.

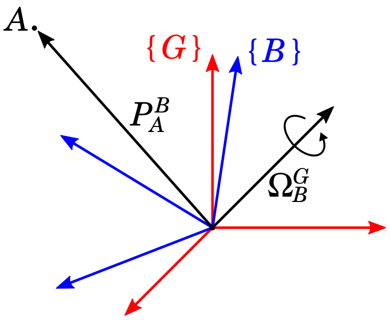


Figure 3.28: The orientation of frame $\{B\}$ relative to frame $\{G\}$ is varying over time.

To do this, we can make use of the fact that the linear velocity of A relative to $\{G\}$ is the cross product of the angular velocity Ω_B^G and the position vector P_A^G (see Fig. 3.29). That is,

$$V_{P_A^G}^G = \Omega_B^G \times P_A^G. \tag{3.193}$$

Since we need a change in frame, we use the appropriate rotation matrix to write

$$V_{P_A^B}^G = \Omega_B^G \times R_B^G P_A^B, \tag{3.194}$$

which holds when we assume that P_A^B does not vary in time.

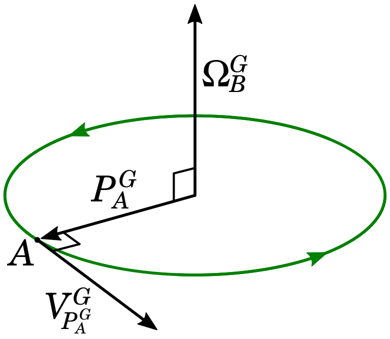


Figure 3.29: The linear velocity of point A relative to frame $\{G\}$ is induced by the angular velocity of frame $\{B\}$ relative to $\{G\}$ and is computed as a cross product.

Simultaneous Linear and Angular Velocity

Now we will remove our previous assumptions to address simultaneous linear and angular velocity. If frame $\{B\}$ is both translating (as in Fig. 3.27) and rotating (as in Fig. 3.28) with respect to frame $\{G\}$, then by adding (3.192) and (3.194) we can derive the general formula for the velocity of a point A located with respect to $\{B\}$ as seen from $\{G\}$:

$$V^G_{P^B_A} = V^G_B + R^G_B V^B_{P^B_A} + \Omega^G_B \times R^G_B P^B_A. \tag{3.195}$$

Another way to interpret (3.195) is that it is the derivative of a vector fixed in a moving frame as seen from a stationary frame.

3.9.3 The Jacobian: Manipulator Arms in Motion

We have the basic tools we need to describe the continuous motion of rigid bodies, so we will begin our study of velocity kinematics for manipulator arms, in which our goal is to relate the linear velocity and angular velocity of the end-effector to the velocities of the joints. In other words, we wish to relate V^0_n and Ω^0_n to $\frac{d}{dt}\mathbf{q}$, where $\mathbf{q} = \begin{bmatrix} q_1 & q_2 & \dots & q_n \end{bmatrix}^T$ is the vector of joint variables.

To determine how we might relate these quantities, let us work out an example in two dimensions. Suppose that we have a planar arm with two revolute joints, as shown in Fig. 3.30. We know from (3.190) that we can compute V^0_n as $\frac{d}{dt}P^0_n$, so let

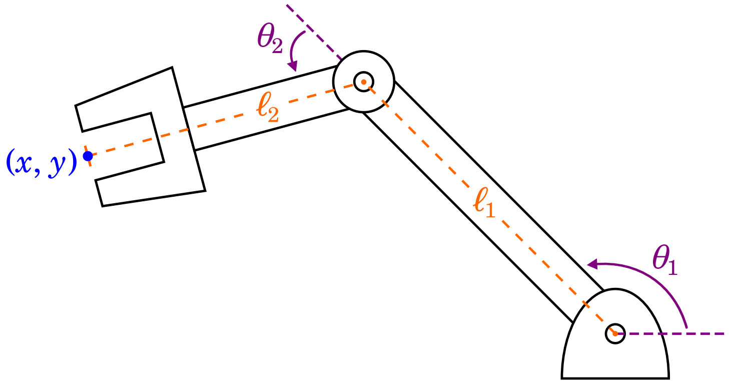


Figure 3.30: A planar manipulator with the origin of the end-effector frame at (x, y).

us work out the forward kinematics for the position of the end-effector, as

$$\begin{aligned}
\begin{bmatrix} x \\ y \end{bmatrix} &= R_{\theta_1}(\begin{bmatrix} \ell_1 & 0 \end{bmatrix}^T) + R_{\theta_1+\theta_2}(\begin{bmatrix} \ell_2 & 0 \end{bmatrix}^T) && (3.196) \\
&= \begin{bmatrix} \ell_1 \cos\theta_1 \\ \ell_1 \sin\theta_1 \end{bmatrix} + \begin{bmatrix} \ell_2 \cos(\theta_1+\theta_2) \\ \ell_2 \sin(\theta_1+\theta_2) \end{bmatrix} && (3.197) \\
&= \begin{bmatrix} \ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1+\theta_2) \\ \ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1+\theta_2) \end{bmatrix}. && (3.198)
\end{aligned}$$

Now we need to differentiate this result with respect to time. Since θ_1 and θ_2 are both functions of time, the forward kinematics equations have the form

$$\begin{aligned}
&x(\theta_1(t), \theta_2(t)), && (3.199) \\
&y(\theta_1(t), \theta_2(t)). && (3.200)
\end{aligned}$$

We therefore need to use the chain rule when differentiating. The result is

$$\begin{bmatrix} \dfrac{dx}{dt} \\ \dfrac{dy}{dt} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial x}{\partial \theta_1}\dfrac{d\theta_1}{dt} + \dfrac{\partial x}{\partial \theta_2}\dfrac{d\theta_2}{dt} \\ \dfrac{\partial y}{\partial \theta_1}\dfrac{d\theta_1}{dt} + \dfrac{\partial y}{\partial \theta_2}\dfrac{d\theta_2}{dt} \end{bmatrix}. \tag{3.201}$$

Our goal is to relate the end-effector velocity to the joint velocities, so we pull the joint velocities out into a separate vector, yielding

$$\begin{bmatrix} \dfrac{dx}{dt} \\ \dfrac{dy}{dt} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial x}{\partial \theta_1} & \dfrac{\partial x}{\partial \theta_2} \\ \dfrac{\partial y}{\partial \theta_1} & \dfrac{\partial y}{\partial \theta_2} \end{bmatrix} \begin{bmatrix} \dfrac{d\theta_1}{dt} \\ \dfrac{d\theta_2}{dt} \end{bmatrix}. \tag{3.202}$$

The 2×2 matrix in (3.202) is a very important matrix called the **Jacobian matrix**[5]. We can think of the Jacobian matrix as a multidimensional form of the derivative. In general, the form of the Jacobian matrix is as follows. Suppose that we have m equations in n variables,

$$\begin{aligned}
p_1 &= f_1(q_1, q_2, \ldots, q_n), \\
p_2 &= f_2(q_1, q_2, \ldots, q_n), \\
&\vdots \\
p_m &= f_m(q_1, q_2, \ldots, q_n),
\end{aligned}$$

[5] The Jacobian matrix is often referred to as “the Jacobian,” which can also mean the determinant of a square Jacobian matrix. In these notes, “the Jacobian” will always mean the Jacobian matrix.

which we may write as a vector function,

$$\mathbf{p} = \mathbf{f}(\mathbf{q}). \tag{3.203}$$

Then the Jacobian matrix of **p** is the matrix of the partial derivatives of **p**:

$$J(q_1, q_2, \ldots, q_n) = \begin{bmatrix} \frac{\partial p_1}{\partial q_1} & \frac{\partial p_1}{\partial q_2} & \cdots & \frac{\partial p_1}{\partial q_n} \\ \frac{\partial p_2}{\partial q_1} & \frac{\partial p_2}{\partial q_2} & \cdots & \frac{\partial p_2}{\partial q_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial p_m}{\partial q_1} & \frac{\partial p_m}{\partial q_2} & \cdots & \frac{\partial p_m}{\partial q_n} \end{bmatrix}. \tag{3.204}$$

In the example we have been working through, we have two equations (x and y) in two variables (θ_1 and θ_2). Hence, as we saw in (3.202), the Jacobian of $\begin{bmatrix} x & y \end{bmatrix}^T$ is

$$J(\theta_1, \theta_2) = \begin{bmatrix} \dfrac{\partial x}{\partial \theta_1} & \dfrac{\partial x}{\partial \theta_2} \\ \dfrac{\partial y}{\partial \theta_1} & \dfrac{\partial y}{\partial \theta_2} \end{bmatrix}. \tag{3.205}$$

To finish our example, we will differentiate with respect to each joint variable, as

$$\frac{\partial}{\partial \theta_1} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -\ell_1 \sin\theta_1 - \ell_2 \sin(\theta_1 + \theta_2) \\ \ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \tag{3.206}$$

$$\frac{\partial}{\partial \theta_2} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -\ell_2 \sin(\theta_1 + \theta_2) \\ \ell_2 \cos(\theta_1 + \theta_2) \end{bmatrix}. \tag{3.207}$$

Now we can write the Jacobian as

$$J_V(\theta_1, \theta_2) = \begin{bmatrix} -\ell_1 \sin\theta_1 - \ell_2 \sin(\theta_1 + \theta_2) & -\ell_2 \sin(\theta_1 + \theta_2) \\ \ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1 + \theta_2) & \ell_2 \cos(\theta_1 + \theta_2) \end{bmatrix}, \tag{3.208}$$

You may have noticed in (3.208) that we added a subscript to our notation for the Jacobian, as J_V. We did this because (3.208) only gives us the linear velocity of the end-effector as a function of the joint variables; it does not tell us anything about the angular velocity. Indeed, throughout this example, we have neglected the angular velocity of the end-effector entirely. Fortunately, it is very simple to compute for planar manipulators, which only have one rotational degree of freedom. The angle of the end-effector is given by the sum of the joint angles, as

$$\phi = \theta_1 + \theta_2. \tag{3.209}$$

We therefore have one equation in two variables, and the Jacobian is

$$J_\Omega(\theta_1, \theta_2) = \begin{bmatrix} \frac{\partial \phi}{\partial \theta_1} & \frac{\partial \phi}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix}. \tag{3.210}$$

Thus, we have

$$\frac{d\phi}{dt} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{d\theta_1}{dt} \\ \frac{d\theta_2}{dt} \end{bmatrix} = \frac{d\theta_1}{dt} + \frac{d\theta_2}{dt}, \tag{3.211}$$

which tells us that the angular velocity of the end-effector frame is the sum of the velocities of the joint angles.

Now we have two Jacobian matrices for the planar arm we have been studying: one for the linear velocity of the end-effector, and one for the angular velocity. Stacking these two matrices gives us the **manipulator Jacobian** of the arm, as

$$J = \begin{bmatrix} J_V \\ J_\Omega \end{bmatrix} = \begin{bmatrix} -\ell_1 \sin\theta_1 - \ell_2 \sin(\theta_1 + \theta_2) & -\ell_2 \sin(\theta_1 + \theta_2) \\ \ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1 + \theta_2) & \ell_2 \cos(\theta_1 + \theta_2) \\ 1 & 1 \end{bmatrix}. \tag{3.212}$$

Depending on the application, the manipulator Jacobian may be referred to as “the Jacobian” for short. Therefore, one should pay attention to whether “the Jacobian” is being used to mean J, J_V, or J_Ω.

Using the manipulator Jacobian, we can relate both the linear velocity and the angular velocity of the end-effector to the velocities of the joints, as

$$\begin{bmatrix} V_n^0 \\ \Omega_n^0 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix},$$

where the overdot notation $\dot{q}$ means $\frac{dq}{dt}$.

The dimensionality of the manipulator Jacobian is important. In this example, J is a 3×2 matrix, which is representative of the fact that the arm has three degrees of freedom (two linear, one angular) and two joints. In general, J will have as many rows as there are degrees of freedom in the Cartesian space being considered and as many columns as there are joints of the arm. We will discuss the importance of the dimensionality more later. First, we will describe a general procedure for deriving the manipulator Jacobian for an arbitrary manipulator in three dimensions.

Step 1: Determining the columns of J_V

Recalling the form of (3.205), we see that the ith column of J_V is given by

$$\frac{\partial}{\partial q_i} P_n^0. \tag{3.213}$$

Now suppose that $q_i = 1$, meaning that joint i is being actuated at unit velocity, and all of the other joints have zero velocity. Then (3.213) is the velocity of the end-effector. We can use this observation to derive the ith column. We must consider two cases: the case of a prismatic joint, and the case of a revolute joint.

Case 1: joint i is prismatic.

A prismatic joint i joint imparts a pure translation on the end-effector that is parallel to the axis $\hat{z}^0_{i-1}$. The rate of translation is $\dot{d}_i$, and thus:

$$V^0_n = \dot{P}^0_n = \hat{z}^0_{i-1}\dot{d}_i. \tag{3.214}$$

Thus, column i of J_V is z^0_{i-1} when joint i is prismatic.

Case 2: joint i is revolute.

Recalling (3.193), we can write the linear velocity of the origin of $\{n\}$ relative to $\{i-1\}$ induced by angular velocity Ω^{i-1}_n as

$$V^{i-1}_n = \Omega^{i-1}_n \times P^{i-1}_n. \tag{3.215}$$

We wish to find V^0_n rather than V^{i-1}_n, however, so we need to express P^{i-1}_n in frame $\{0\}$. We can achieve this through vector subtraction, as $P^0_n - P^0_{i-1}$. Thus,

$$V^0_n = \Omega^{i-1}_n \times (P^0_n - P^0_{i-1}). \tag{3.216}$$

Since $\Omega^{i-1}_n = \hat{z}^0_{i-1}\dot{\theta}_i$, we can conclude that column i of J_V is

$$\hat{z}^0_{i-1} \times (P^0_n - P^0_{i-1}) \tag{3.217}$$

when joint i is revolute.

Step 2: Determining the columns of J_Ω

Now that we have done the work to determine J_V, it is straightforward to determine each column of J_Ω. We consider each case below.

Case 1: joint i is prismatic.

As previously mentioned, a prismatic joint i imparts a pure translation on the end-effector. As a result, the actuation of joint i does not induce any angular velocity, and therefore column i of J_Ω is **0** when joint i is prismatic.

Case 2: joint i is revolute.

We saw previously that actuating a revolute joint i induces an angular velocity of $\hat{z}^0_{i-1}\dot{\theta}_i$ on the end-effector. Thus, the ith column of J_Ω is $\hat{z}^0_{i-1}$ for a revolute joint i.

Putting it all together

Let us return to our example of the planar arm with two revolute joints. Using the method described above, we have

$$J = \begin{bmatrix} \hat{z}_0^0 \times (P_2^0 - P_0^0) & \hat{z}_0^0 \times (P_2^0 - P_1^0) \\ \hat{z}_0^0 & \hat{z}_1^0 \end{bmatrix}. \tag{3.218}$$

Like before, we inspect Fig. 3.30 to determine that

$$P_0^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \ P_1^0 = \begin{bmatrix} \ell_1 \cos\theta_1 \\ \ell_1 \sin\theta_1 \\ 0 \end{bmatrix}, \ P_2^0 = \begin{bmatrix} \ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1+\theta_2) \\ \ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1+\theta_2) \\ 0 \end{bmatrix}, \tag{3.219}$$

and since the arm is planar, we have

$$\hat{z}_0^0 = \hat{z}_1^0 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T. \tag{3.220}$$

Thus, we have

$$J = \begin{bmatrix} -\ell_1 \sin\theta_1 - \ell_2 \sin(\theta_1+\theta_2) & -\ell_2 \sin(\theta_1+\theta_2) \\ \ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1+\theta_2) & \ell_2 \cos(\theta_1+\theta_2) \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}. \tag{3.221}$$

Note that this 6×2 Jacobian has the same three rows as our previous 3×2 Jacobian from (3.212) along with three rows that are all zeros. Intuitively, this makes sense because our planar arm only has three degrees of freedom, but we are considering Cartesian space with six degrees of freedom.

Inverse Velocity Kinematics

Now that we can compute the linear and angular velocity of the end-effector given the manipulator Jacobian and the velocities of the joints, it is natural to ask: can we go the other direction? In other words, given a desired linear and angular velocity of the end-effector and a joint configuration $\mathbf{q}$, can we determine the joint velocities that will achieve the desired end-effector linear and angular velocity? As we will later see, solving this problem of inverse velocity kinematics is a very useful tool for numerically computing inverse kinematics.

From an analytic standpoint, it is very easy to state this inverse relationship. Suppose that we are interested in specifying a desired linear velocity for the end-effector of our planar manipulator with two revolute joints. Recall that

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J_V \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}. \tag{3.222}$$

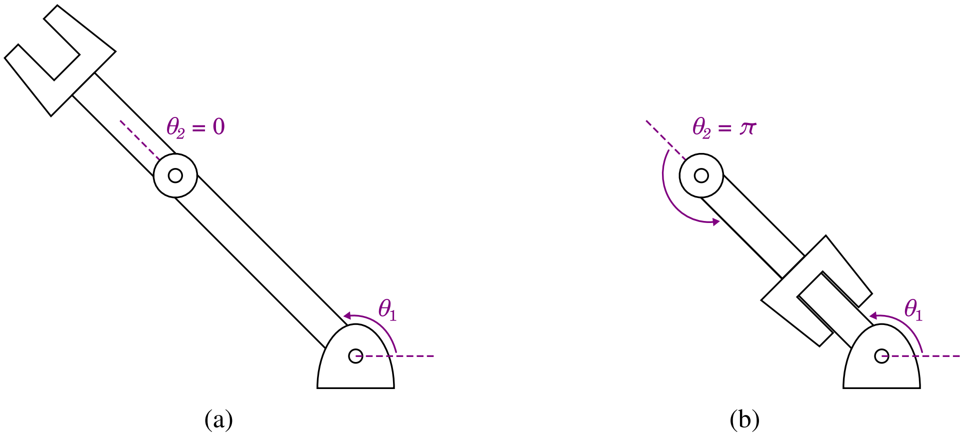
The inverse of this relationship is given as

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = J_V^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}. \tag{3.223}$$

Therefore, to carry out the computation, we need to invert J_V, as

$$J_V^{-1} = \frac{1}{\ell_1 \ell_2 \sin\theta_2} \begin{bmatrix} \ell_2 \cos(\theta_1+\theta_2) & \ell_2 \sin(\theta_1+\theta_2) \\ -\ell_1 \cos\theta_1 - \ell_2 \cos(\theta_1+\theta_2) & -\ell_1 \sin\theta_1 - \ell_2 \sin(\theta_1+\theta_2) \end{bmatrix}. \tag{3.224}$$

Immediately we can see that a couple problems might arise with this computation. If $\theta_2 = 0$ or $\theta_2 = \pi$, then the determinant goes to zero, and J_V will not have an inverse. These are the two joint angles that result in our planar manipulator being in a *singular configuration* in which a degree of freedom is lost. Geometrically, these are the cases where the arm is either fully extended or completely bent backward on itself (see Fig. 3.31). Since there are infinitesimal motions that the end-effector cannot achieve when in a singular configuration due to having fewer degrees of freedom, we try to avoid these configurations when planning manipulator motions.



(a) (b)

Figure 3.31: Singular configurations of the planar manipulator with two revolute joints.

The problems do not end with singular configurations, however. Suppose that we were interested in specifying both a linear velocity and angular velocity for the end-effector. In this case, we would need to invert either the 3×2 manipulator Jacobian or the 6×2 manipulator Jacobian that we derived, and as we know from linear algebra, we cannot invert a matrix that is not square. When we are faced with this problem, we need to choose some method of approximating the inverse. One option is to approximate J^{-1} with a pseudoinverse J^{+}. Another option is to approximate J^{-1} with the transpose J^{T}. At first, this latter option may seem quite surprising and rather foolish because in general, J^{T} is a very poor approximation of J^{-1}. However, the use of the transpose in place of the inverse will turn out to be valuable in numerical methods that iteratively approximate J^{-1}. In such cases, what allows us to use the transpose in place of the inverse is the fact that through the process of iterating, much of the error introduced by the transpose eventually cancels out. The quality of the solution might not be as good as one computed with a pseudoinverse, but determining the transpose is much less computationally expensive than determining a pseudoinverse.

We will now describe an iterative algorithm for numerically computing inverse kinematics through the use of inverse velocity kinematics. First, it is important to understand why the algorithm needs to be iterative in the first place. The reason is that the Jacobian is a *linearization* of the forward kinematics of a manipulator arm, which are typically complex and very nonlinear. As a result, the relationship that the Jacobian gives us is only instantaneous. Thus, to move the end-effector to a goal pose, we need to use an iterative method, taking small steps toward the goal and computing an approximation of J^{-1} with each step.

The process is as follows. Suppose that the pose of our end-effector is given by the vector $\mathbf{e}$, and the joint variables of the manipulator are given by the vector $\mathbf{q}$. We wish to compute the joint velocities required to move the end-effector to goal pose $\mathbf{g}$. To do this, we repeat the following steps until $\mathbf{e}$ is satisfactorily close to $\mathbf{g}$:

1. Compute $J(\mathbf{q})$.

2. Select an increment that will move $\mathbf{e}$ closer to $\mathbf{g}$, as $\Delta\mathbf{e} = \beta(\mathbf{g} - \mathbf{e})$, where $0 < \beta \leq 1$.

3. Compute the change in the joints variables that will achieve the end-effector increment selected in the previous step, as $\Delta\mathbf{q} = J^{-1}\Delta\mathbf{e}$, using our chosen method of approximating J^{-1}.

4. Apply this change to the joint variable vector, as $\mathbf{q} = \mathbf{q} + \Delta\mathbf{q}$.

5. Update $\mathbf{e}$ by computing the forward kinematics of the manipulator with the updated $\mathbf{q}$.

In addition to choosing a method for approximating J^{-1}, we also need to decide what constitutes $\mathbf{e}$ being "satisfactorily close" to $\mathbf{g}$ before we can implement this algorithm. In other words, how do we know when to stop iterating? In some cases, the algorithm will find a solution, or it will find a solution that is "satisfactorily close" as defined by some tolerance. In other cases, the algorithm might get stuck in a local minimum, or it might take an unreasonably long time. Some methods of addressing these latter cases include limiting the amount of times the algorithm can iterate and starting over with a randomized pose vector.

Finally, it is worth noting that the dimensionality of the Jacobian tells us more than just whether the matrix is invertible or not. In the case where there are more columns than there are rows, we know that the robot must be *redundant*; that is, it has more degrees of freedom than is necessary for performing a specified task. The human arm is an example of a redundant manipulator: while only six degrees of freedom are required to position and orient the hand in any arbitrary way, the human arm has seven degrees of freedom. This additional degree of freedom allows us to move our arm — specifically, reposition our elbow — while keeping our hand in the exact same position and orientation. (Prove this to yourself: plant your hand on the surface of a table and then move your elbow around.)

When a robot is redundant, the columns of its Jacobian are linearly dependent. (To see why, consider that there cannot be more than six linearly independent vectors in $\mathbb{R}^6$, so the seven columns of the Jacobian matrix for a human arm must be linearly dependent.) It follows that the Jacobian matrix has a nontrivial null space. Velocities drawn from the null space generate internal joint motions but do not cause any motion at the end-effector. Thus, when we are solving the inverse kinematics problem, we can use the null space of the manipulator Jacobian to optimize a solution or find an alternate trajectory.

3.9.4 Long-Form Examples

Example 3.8 Forward Velocity Kinematics
Concepts reviewed: *forward kinematics*, *the Jacobian*
Problem: Suppose you have a two-link robot arm equipped with a billiards cue, as shown in Fig. 3.32. When the cue strikes the ball, the joints are instantaneously moving at $\dot{\theta}_1 = -1$ rad/sec and $\dot{\theta}_2 = 1$ rad/sec. The instantaneous joint angles are $\theta_1 = \frac{8\pi}{15}$ rad and $\theta_2 = -\frac{\pi}{2}$ rad. The cue strikes the ball with a line of force that passes through its center (i.e. there is no spin, and it goes in a straight line in the direction the cue tip was moving). The cue ball starts exactly centered on a table of dimensions 2 m long by 1 m wide, and the robot link lengths are $\ell_1 = 0.5$ m and $\ell_2 = 0.5$ m. Where will the ball strike the rail at the edge of the table?
Solution: During the collision between cue and ball, the impulse instantly imparts the same velocity on the ball as the cue had at that moment. Therefore, we must find the linear (Cartesian) velocity of the cue tip at that moment.

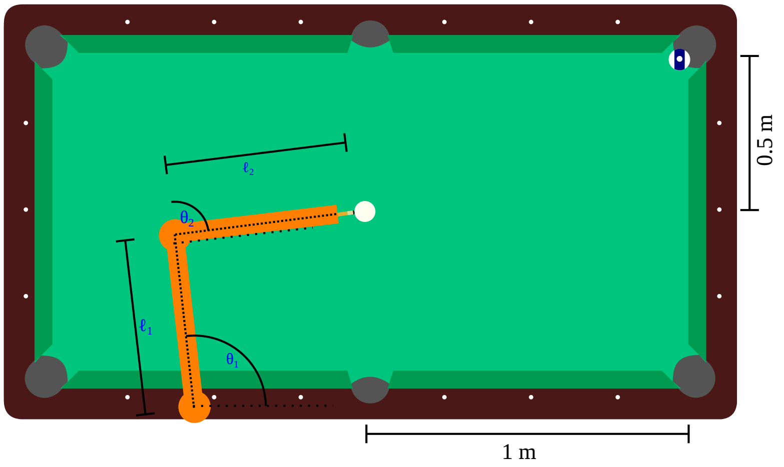


Figure 3.32: A pool table equipped with a robot cue stick. The robot always strikes the ball at its center to avoid spin.

We can use J_V from (3.208) since the arm has the same geometry as the one in Fig. 3.30:

$$\begin{bmatrix}\dot{x}_2\\ \dot{y}_2\end{bmatrix} = J_V \begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{bmatrix} \tag{3.225}$$

$$= \begin{bmatrix}-\ell_1\sin\theta_1 - \ell_2\sin(\theta_1+\theta_2) & -\ell_2\sin(\theta_1+\theta_2)\\ \ell_1\cos\theta_2 + \ell_2\cos(\theta_1+\theta_2) & \ell_2\cos(\theta_1+\theta_2)\end{bmatrix}\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{bmatrix} \tag{3.226}$$

$$= \begin{bmatrix}-\ell_1\sin\theta_1 - (\ell_2\dot{\theta}_1 - \ell_2\dot{\theta}_2)\sin(\theta_1+\theta_2)\\ \ell_1\cos\theta_1 + (\ell_2\dot{\theta}_1 + \ell_2\dot{\theta}_2)\cos(\theta_1+\theta_2)\end{bmatrix} \tag{3.227}$$

$$\approx \begin{bmatrix}0.497261\\ 0.052264\end{bmatrix} \tag{3.228}$$

Computing the slope $m = \frac{\dot{y}_2}{\dot{x}_2} \approx 0.10510$, we can predict that the cue ball will hit the side rail about 10.5 cm above the center. □

Example 3.9 Inverse Velocity Kinematics
Concepts reviewed: *inverse kinematics*, *the Jacobian*
Problem: Again considering the two-link robot arm with the billiards cue in Fig. 3.32, what arm velocity parameters does the robot need to set in order to strike the ball in the top-right corner pocket with the cue ball? Assume that the cue ball will travel at 10 m/s and that the joint angles at the moment of impact remain $\theta_1 = \frac{8\pi}{15}$ and $\theta_2 = -\frac{\pi}{2}$.

Solution: The desired direction of the ball can be derived from the diagram as $\phi = \text{atan2}(0.5, 1) \approx 0.46365$. Scaled by the desired magnitude, we have

$$\begin{bmatrix} \dot{x_2} \\ \dot{y_2} \end{bmatrix} = \begin{bmatrix} 10\cos(\phi) \\ 10\sin(\phi) \end{bmatrix} \approx \begin{bmatrix} 8.9443 \\ 4.4721 \end{bmatrix}. \tag{3.229}$$

The values for $\theta_1 = \frac{8\pi}{15}$ and $\theta_2 = -\frac{\pi}{2}$ are taken from Example 3.8 because they solve the inverse kinematics problem. Since J_V is square and has full rank, we can compute its inverse, as

$$J_V^{-1} = \frac{1}{\ell_1\ell_2\sin(\theta_2)} \begin{bmatrix} \ell_2\cos(\theta_1+\theta_2) & \ell_2\sin(\theta_1+\theta_2) \\ -\ell_1\cos\theta_1 - \ell_2\cos(\theta_1+\theta_2) & -\ell_1\sin\theta_1 - \ell_2\sin(\theta_1+\theta_2) \end{bmatrix} \tag{3.230}$$

$$\approx \begin{bmatrix} -1.9890 & -0.2091 \\ 1.7800 & 2.1981 \end{bmatrix}, \tag{3.231}$$

and then solve for the θ rates, yielding

$$\begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \end{bmatrix} = J_V^{-1} \begin{bmatrix} \dot{x_2} \\ \dot{y_2} \end{bmatrix} \tag{3.232}$$

$$\approx \begin{bmatrix} -1.9890 & -0.2091 \\ 1.7800 & 2.1981 \end{bmatrix} \begin{bmatrix} 8.9443 \\ 4.4721 \end{bmatrix} \tag{3.233}$$

$$\approx \begin{bmatrix} -18.725 \\ 25.751 \end{bmatrix}. \tag{3.234}$$

□

3.10 Further Reading

For more discussion of kinematics, see Lynch and Park [2] (available online), Craig [1], Murray et al. [3] and Spong et al. [4].

Bibliography

[1] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1989. ISBN 0201095289.

[2] K. M. Lynch and F. C. Park. *Modern Robotics*. Cambridge University Press, 2017.

[3] R. M. Murray, S. S. Sastry, and L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994. ISBN 0849379814.

[4] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. John Wiley & Sons, Hoboken (N.J.), 2006. ISBN 0-471-64990-2.