Chapter 13

Wheeled Mobile Robots

A kinematic model of a mobile robot governs how wheel speeds map to robot velocities, while a dynamic model governs how wheel torques map to robot accelerations. In this chapter, we ignore the dynamics and focus on the kinematics. We also assume that the robots roll on hard, flat, horizontal ground without skidding (i.e., tanks and skid-steered vehicles are excluded). The mobile robot is assumed to have a single rigid-body chassis (not articulated like a tractor-trailer) with a configuration $T_{sb} \in SE(2)$ representing a chassis-fixed frame {b} relative to a fixed space frame {s} in the horizontal plane. We represent T_{sb} by the three coordinates $q = (\phi, x, y)$. We also usually represent the velocity of the chassis as the time derivative of the coordinates, $\dot{q} = (\dot{\phi}, \dot{x}, \dot{y})$. Occasionally it will be convenient to refer to the chassis' planar twist $\mathcal{V}_b = (\omega_{bz}, v_{bx}, v_{by})$ expressed in {b}, where

$$\mathcal{V}_b = \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix}, \tag{13.1}$$

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix}. \tag{13.2}$$

This chapter covers kinematic modeling, motion planning, and feedback control for wheeled mobile robots, and concludes with a brief introduction to mobile manipulation, which is the problem of controlling the end-effector motion of a robot arm mounted on a mobile platform.

Figure 13.1: (Left) A typical wheel that rolls without sideways slip – here a unicycle wheel. (Middle) An omniwheel. (Right) A mecanum wheel. Omniwheel and mecanum wheel images from VEX Robotics, Inc., used with permission.

13.1 Types of Wheeled Mobile Robots

Wheeled mobile robots may be classified in two major categories, **omnidirectional** and **nonholonomic**. Omnidirectional mobile robots have no equality constraints on the chassis velocity $\dot{q} = (\dot{\phi}, \dot{x}, \dot{y})$, while nonholonomic robots are subject to a single Pfaffian velocity constraint $A(q)\dot{q} = 0$ (see Section 2.4 for an introduction to Pfaffian constraints). For a car-like robot, this constraint prevents the car from moving directly sideways. Despite this velocity constraint, the car can reach any (ϕ, x, y) configuration in an obstacle-free plane. In other words, the velocity constraint cannot be integrated to an equivalent configuration constraint, and therefore it is a nonholonomic constraint.

Whether a wheeled mobile robot is omnidirectional or nonholonomic depends in part on the type of wheels it employs (Figure 13.1). Nonholonomic mobile robots employ conventional wheels, such as you might find on your car: the wheel rotates about an axle perpendicular to the plane of the wheel at the wheel's center, and optionally it can be steered by spinning the wheel about an axis perpendicular to the ground at the contact point. The wheel rolls without sideways slip, which is the source of the nonholonomic constraint on the robot's chassis.

Omnidirectional wheeled mobile robots typically employ either **omniwheels** or **mecanum wheels**.[1] An omniwheel is a typical wheel augmented with rollers on its outer circumference. These rollers spin freely about axes in the plane of

[1] These types of wheels are often called "Swedish wheels," as they were invented by Bengt Ilon working at the Swedish company Mecanum AB. The usage of, and the differentiation between, the terms "omniwheel," "mecanum wheel," and "Swedish wheel" is not completely standard, but here we use one popular choice.

the wheel and tangential to the wheel's outer circumference, and they allow sideways sliding while the wheel drives forward or backward without slip in that direction. Mecanum wheels are similar, except that the spin axes of the circumferential rollers are not in the plane of the wheel (see Figure 13.1). The sideways sliding allowed by omniwheels and mecanum wheels ensures that there are no velocity constraints on the robot's chassis.

Omniwheels and mecanum wheels are not steered, only driven forward or backward. Because of their small diameter rollers, omniwheels and mecanum wheels work best on hard, flat ground.

The issues in the modeling, motion planning, and control of wheeled mobile robots depend intimately on whether the robot is omnidirectional or nonholonomic, so we treat these two cases separately in the following sections.

13.2 Omnidirectional Wheeled Mobile Robots

13.2.1 Modeling

An omnidirectional mobile robot must have at least three wheels to achieve an arbitrary three-dimensional chassis velocity $\dot{q} = (\dot{\phi}, \dot{x}, \dot{y})$, since each wheel has only one motor (controlling its forward–backward velocity). Figure 13.2 shows two omnidirectional mobile robots, one with three omniwheels and one with four mecanum wheels. Also shown are the wheel motions obtained by driving the wheel motors as well as the free sliding motions allowed by the rollers.

Two important questions in kinematic modeling are the following.

(a) Given a desired chassis velocity $\dot{q}$, at what speeds must the wheels be driven?

(b) Given limits on the individual wheel driving speeds, what are the limits on the chassis velocity $\dot{q}$?

To answer these questions, we need to understand the wheel kinematics illustrated in Figure 13.3. In a frame $\hat{x}_w$–$\hat{y}_w$ at the center of the wheel, the linear velocity of the center of the wheel is written $v = (v_x, v_y)$, which satisfies

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = v_{\text{drive}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + v_{\text{slide}} \begin{bmatrix} -\sin\gamma \\ \cos\gamma \end{bmatrix}, \tag{13.3}$$

where γ denotes the angle at which free "sliding" occurs (allowed by the passive rollers on the circumference of the wheel), v_{drive} is the driving speed, and v_{slide} is

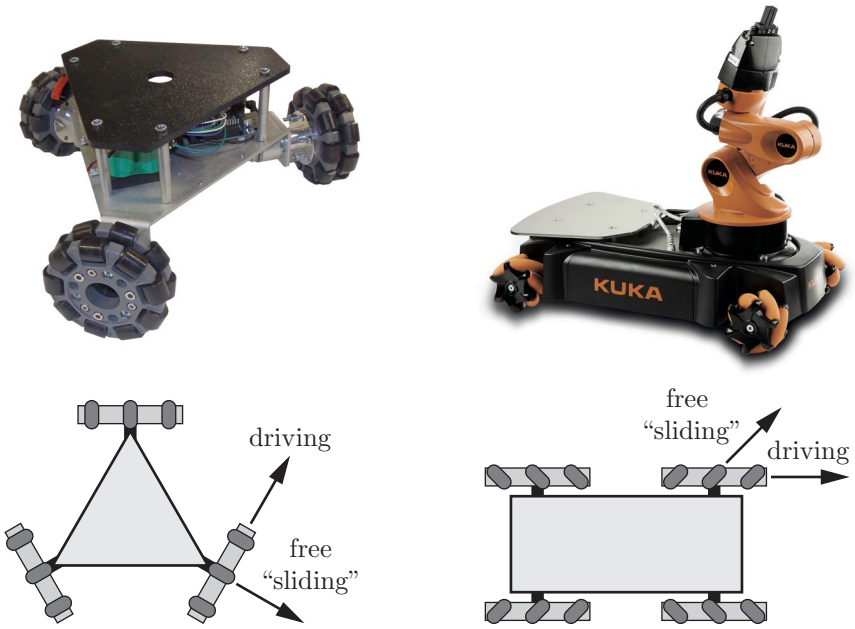

Figure 13.2: (Left) A mobile robot with three omniwheels. Also shown for one omniwheel is the direction in which the wheel can freely slide due to the rollers, as well as the direction in which the wheel rolls without slipping when driven by the wheel motor. (The upper image is from www.superdroidrobots.com, used with permission.) (Right) The KUKA youBot mobile manipulator system, which uses four mecanum wheels for its mobile base. (The upper image is from KUKA Roboter GmbH, used with permission.)

the sliding speed. For an omniwheel $\gamma = 0$ and, for a mecanum wheel, typically $\gamma = \pm 45°$. Solving Equation (13.3), we get

$$v_{\text{drive}} = v_x + v_y \tan\gamma,$$
$$v_{\text{slide}} = v_y / \cos\gamma.$$

Letting r be the radius of the wheel and u be the driving angular speed of the wheel,

$$u = \frac{v_{\text{drive}}}{r} = \frac{1}{r}(v_x + v_y \tan\gamma). \tag{13.4}$$

To derive the full transformation from the chassis velocity $\dot{q} = (\dot{\phi}, \dot{x}, \dot{y})$ to the driving angular speed u_i for wheel i, refer to the notation illustrated in

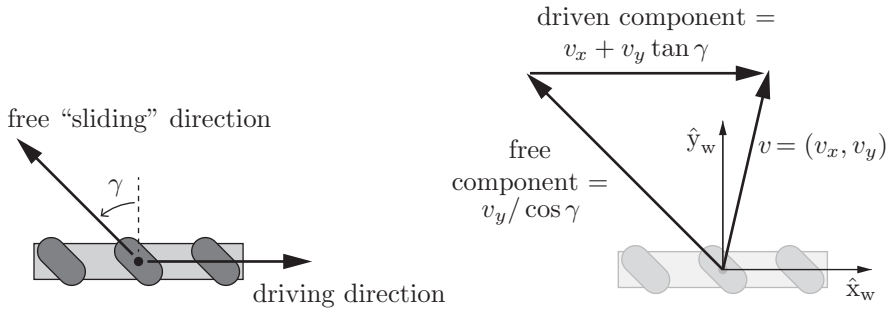

Figure 13.3: (Left) The driving direction and the direction in which the rollers allow the wheel to slide freely. For an omniwheel $\gamma = 0$ and, for a mecanum wheel, typically $\gamma = \pm 45°$. (Right) The driven and free sliding speeds for the wheel velocity $v = (v_x, v_y)$ expressed in the wheel frame $\hat{x}_w$–$\hat{y}_w$, where the $\hat{x}_w$-axis is aligned with the forward driving direction.

Figure 13.4. The chassis frame {b} is at $q = (\phi, x, y)$ in the fixed space frame {s}. The center of the wheel and its driving direction are given by (β_i, x_i, y_i) expressed in {b}, the wheel's radius is r_i, and the wheel's sliding direction is given by γ_i. Then u_i is related to $\dot{q}$ by

$$u_i = h_i(\phi)\dot{q} = \begin{bmatrix} \frac{1}{r_i} & \frac{\tan\gamma_i}{r_i} \end{bmatrix} \begin{bmatrix} \cos\beta_i & \sin\beta_i \\ -\sin\beta_i & \cos\beta_i \end{bmatrix} \begin{bmatrix} -y_i & 1 & 0 \\ x_i & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix}. \tag{13.5}$$

Reading from right to left: the first transformation expresses $\dot{q}$ as $\mathcal{V}_b$; the second transformation produces the linear velocity at the wheel in {b}; the third transformation expresses this linear velocity in the wheel frame $\hat{x}_w$–$\hat{y}_w$; and the final transformation calculates the driving angular velocity using Equation (13.4).

Evaluating Equation (13.5) for $h_i(\phi)$, we get

$$h_i(\phi) = \frac{1}{r_i \cos\gamma_i} \begin{bmatrix} x_i \sin(\beta_i + \gamma_i) - y_i \cos(\beta_i + \gamma_i) \\ \cos(\beta_i + \gamma_i + \phi) \\ \sin(\beta_i + \gamma_i + \phi) \end{bmatrix}^{\mathrm{T}}. \tag{13.6}$$

For an omnidirectional robot with $m \geq 3$ wheels, the matrix $H(\phi) \in \mathbb{R}^{m \times 3}$ mapping a desired chassis velocity $\dot{q} \in \mathbb{R}^3$ to the vector of wheel driving speeds

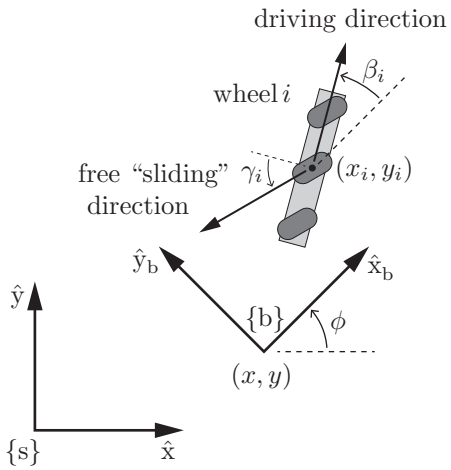

Figure 13.4: The fixed space frame {s}, a chassis frame {b} at (ϕ, x, y) in {s}, and wheel i at (x_i, y_i) with driving direction β_i, both expressed in {b}. The sliding direction of wheel i is defined by γ_i.

$u \in \mathbb{R}^m$ is constructed by stacking the m rows $h_i(\phi)$:

$$u = H(\phi)\dot{q} = \begin{bmatrix} h_1(\phi) \\ h_2(\phi) \\ \vdots \\ h_m(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix}. \tag{13.7}$$

We can also express the relationship between u and the body twist $\mathcal{V}_b$. This mapping does not depend on the chassis orientation ϕ:

$$u = H(0)\mathcal{V}_b = \begin{bmatrix} h_1(0) \\ h_2(0) \\ \vdots \\ h_m(0) \end{bmatrix} \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix}. \tag{13.8}$$

The wheel positions and headings (β_i, x_i, y_i) in {b}, and their free sliding directions γ_i, must be chosen so that $H(0)$ is rank 3. For example, if we constructed a mobile robot of omniwheels whose driving directions and sliding directions were all aligned, the rank of $H(0)$ would be 2, and there would be no way to controllably generate translational motion in the sliding direction.

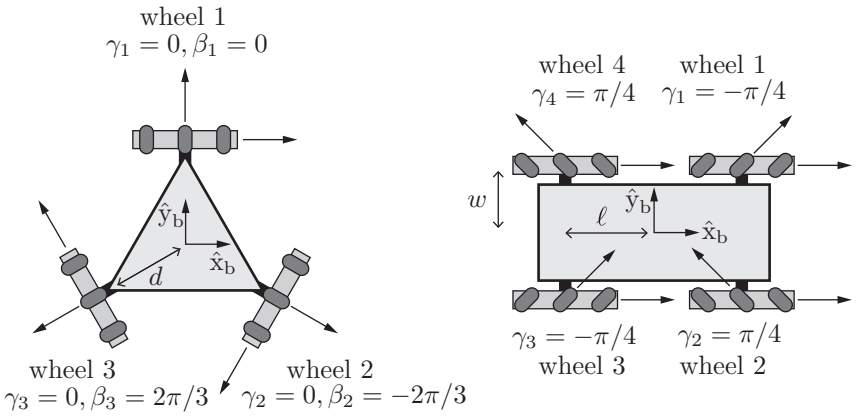

Figure 13.5: Kinematic models for mobile robots with three omniwheels and four mecanum wheels. The radius of all wheels is r and the driving direction for each of the mecanum wheels is $\beta_i = 0$.

In the case $m > 3$, as for the four-wheeled youBot of Figure 13.2, choosing u such that Equation (13.8) is not satisfied for any $\mathcal{V}_b \in \mathbb{R}^3$ implies that the wheels must skid in their driving directions.

Using the notation in Figure 13.5, the kinematic model of the mobile robot with three omniwheels is

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = H(0)\mathcal{V}_b = \frac{1}{r} \begin{bmatrix} -d & 1 & 0 \\ -d & -1/2 & -\sin(\pi/3) \\ -d & -1/2 & \sin(\pi/3) \end{bmatrix} \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix} \tag{13.9}$$

and the kinematic model of the mobile robot with four mecanum wheels is

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = H(0)\mathcal{V}_b = \frac{1}{r} \begin{bmatrix} -\ell - w & 1 & -1 \\ \ell + w & 1 & 1 \\ \ell + w & 1 & -1 \\ -\ell - w & 1 & 1 \end{bmatrix} \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix}. \tag{13.10}$$

For the mecanum robot, to move in the direction $+\hat{x}_b$, all wheels drive forward at the same speed; to move in the direction $+\hat{y}_b$, wheels 1 and 3 drive backward and wheels 2 and 4 drive forward at the same speed; and to rotate in the counterclockwise direction, wheels 1 and 4 drive backward and wheels 2 and 3 drive forward at the same speed. Note that the robot chassis is capable of the same speeds in the forward and sideways directions.

If the driving angular velocity of wheel i is subject to the bound $|u_i| \leq u_{i,\max}$, i.e.,

$$-u_{i,\max} \leq u_i = h_i(0)\mathcal{V}_b \leq u_{i,\max},$$

then two parallel constraint planes defined by $-u_{i,\max} = h_i(0)\mathcal{V}_b$ and $u_{i,\max} = h_i(0)\mathcal{V}_b$ are generated in the three-dimensional space of body twists. Any $\mathcal{V}_b$ between these two planes does not violate the maximum driving speed of wheel i, while any $\mathcal{V}_b$ outside this slice is too fast for wheel i. The normal direction to the constraint planes is $h_i^{\mathrm{T}}(0)$, and the points on the planes closest to the origin are $-u_{i,\max}h_i^{\mathrm{T}}(0)/\|h_i(0)\|^2$ and $u_{i,\max}h_i^{\mathrm{T}}(0)/\|h_i(0)\|^2$.

If the robot has m wheels then the region of feasible body twists V is bounded by the m pairs of parallel constraint planes. The region V is therefore a convex three-dimensional polyhedron. The polyhedron has $2m$ faces and the origin (corresponding to zero twist) is in the center. Visualizations of the six-sided and eight-sided regions V for the three-wheeled and four-wheeled models in Figure 13.5 are shown in Figure 13.6.

13.2.2 Motion Planning

Since omnidirectional mobile robots are free to move in any direction, any of the trajectory planning methods for kinematic systems in Chapter 9, and most of the motion planning methods of Chapter 10, can be adapted.

13.2.3 Feedback Control

Given a desired trajectory $q_d(t)$, we can adopt the feedforward plus PI feedback controller (11.15) to track the trajectory:

$$\dot{q}(t) = \dot{q}_d(t) + K_p(q_d(t) - q(t)) + K_i \int_0^t (q_d(\mathrm{t}) - q(\mathrm{t}))\, d\mathrm{t}, \tag{13.11}$$

where $K_p = k_p I \in \mathbb{R}^{3\times 3}$ and $K_i = k_i I \in \mathbb{R}^{3\times 3}$ have positive values along the diagonal and $q(t)$ is an estimate of the actual configuration derived from sensors. Then $\dot{q}(t)$ can be converted to the commanded wheel driving velocities $u(t)$ using Equation (13.7).

13.3 Nonholonomic Wheeled Mobile Robots

In Section 2.4, the k Pfaffian velocity constraints acting on a system with configuration $q \in \mathbb{R}^n$ were written as $A(q)\dot{q} = 0$, where $A(q) \in \mathbb{R}^{k\times n}$. Instead of specifying the k directions in which velocities are not allowed, we can write the

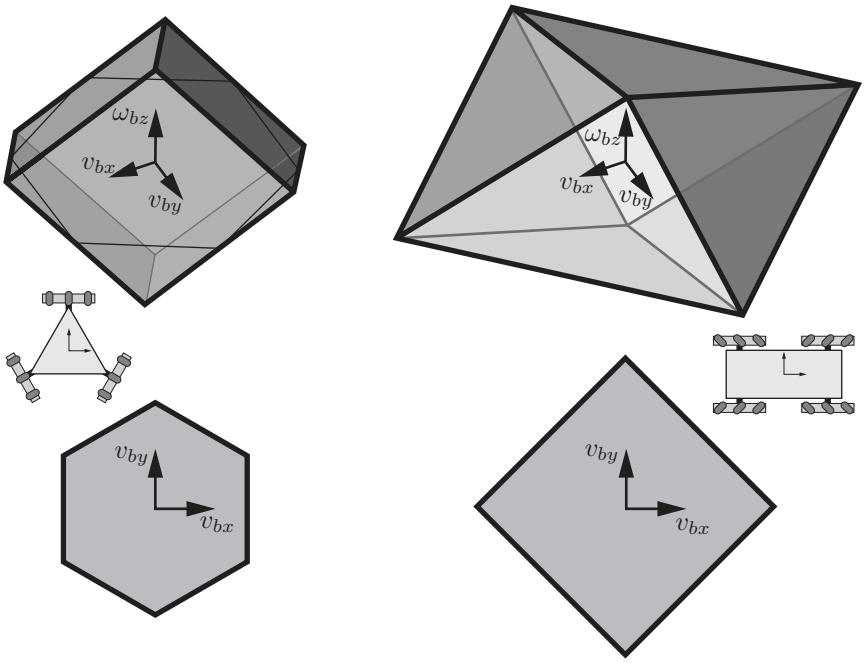

Figure 13.6: (Top row) Regions of feasible body twists V for the three-wheeled (left) and four-wheeled (right) robots of Figure 13.5. Also shown for the three-wheeled robot is the intersection with the $\omega_{bz} = 0$ plane. (Bottom row) The bounds in the $\omega_{bz} = 0$ plane (translational motions only).

allowable velocities of a kinematic system as a linear combination of $n-k$ velocity directions. This representation is equivalent, and it has the advantage that the coefficients of the linear combinations are precisely the controls available to us. We will see this representation in the kinematic models below.

The title of this section implies that the velocity constraints are not integrable to equivalent configuration constraints. We will establish this formally in Section 13.3.2.

13.3.1 Modeling

13.3.1.1 The Unicycle

The simplest wheeled mobile robot is a single upright rolling wheel, or unicycle. Let r be the radius of the wheel. We write the configuration of the wheel as

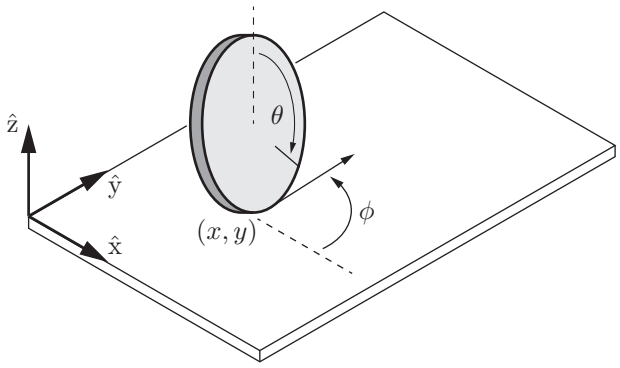

Figure 13.7: A wheel rolling on a plane without slipping.

$q = (\phi, x, y, \theta)$, where (x, y) is the contact point, ϕ is the heading direction, and θ is the rolling angle of the wheel (Figure 13.7). The configuration of the "chassis" (e.g., the seat of the unicycle) is (ϕ, x, y). The kinematic equations of motion are

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ r\cos\phi & 0 \\ r\sin\phi & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = G(q)u = g_1(q)u_1 + g_2(q)u_2. \quad (13.12)$$

The control inputs are $u = (u_1, u_2)$, with u_1 the wheel's forward–backward driving speed and u_2 the heading direction turning speed. The controls are subject to the constraints $-u_{1,\max} \leq u_1 \leq u_{1,\max}$ and $-u_{2,\max} \leq u_2 \leq u_{2,\max}$.

The vector-valued functions $g_i(q) \in \mathbb{R}^4$ are the columns of the matrix $G(q)$, and they are called the **tangent vector fields** (also called the **control vector fields** or simply the **velocity vector fields**) over q associated with the controls $u_i = 1$. Evaluated at a specific configuration q, $g_i(q)$ is a **tangent vector** (or velocity vector) of the tangent vector field.

An example of a vector field on $\mathbb{R}^2$ is illustrated in Figure 13.8.

All our kinematic models of nonholonomic mobile robots will have the form $\dot{q} = G(q)u$, as in Equation (13.12). Three things to notice about these models are: (1) there is no drift – zero controls mean zero velocity; (2) the vector fields $g_i(q)$ are generally functions of the configuration q; and (3) $\dot{q}$ is linear in the controls.

Since we are not usually concerned with the rolling angle of the wheel, we

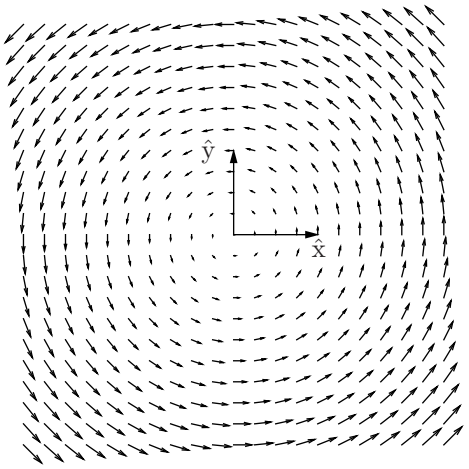

Figure 13.8: The vector field $(\dot{x}, \dot{y}) = (-y, x)$.

can drop the fourth row from (13.12) to get the simplified control system

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ r\cos\phi & 0 \\ r\sin\phi & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{13.13}$$

13.3.1.2 The Differential-Drive Robot

The **differential-drive robot**, or **diff-drive**, is perhaps the simplest wheeled mobile robot architecture. A diff-drive robot consists of two independently driven wheels of radius r that rotate about the same axis, as well as one or more caster wheels, ball casters, or low-friction sliders that keep the robot horizontal. Let the distance between the driven wheels be $2d$ and choose the (x, y) reference point halfway between the wheels (Figure 13.9). Writing the configuration as $q = (\phi, x, y, \theta_{\mathrm{L}}, \theta_{\mathrm{R}})$, where θ_{L} and θ_{R} are the rolling angles of the left and right wheels, respectively, the kinematic equations are

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \\ \dot{\theta}_{\mathrm{L}} \\ \dot{\theta}_{\mathrm{R}} \end{bmatrix} = \begin{bmatrix} -r/2d & r/2d \\ \frac{r}{2}\cos\phi & \frac{r}{2}\cos\phi \\ \frac{r}{2}\sin\phi & \frac{r}{2}\sin\phi \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_{\mathrm{L}} \\ u_{\mathrm{R}} \end{bmatrix}, \tag{13.14}$$

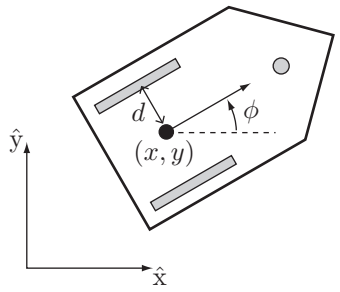

Figure 13.9: A diff-drive robot consisting of two typical wheels and one ball caster wheel, shaded gray.

where u_L is the angular speed of the left wheel and u_R that of the right. A positive angular speed of each wheel corresponds to forward motion at that wheel. The control value at each wheel is taken from the interval $[-u_{max}, u_{max}]$.

Since we are not usually concerned with the rolling angles of the two wheels, we can drop the last two rows to get the simplified control system

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -r/2d & r/2d \\ \frac{r}{2}\cos\phi & \frac{r}{2}\cos\phi \\ \frac{r}{2}\sin\phi & \frac{r}{2}\sin\phi \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix}. \tag{13.15}$$

Two advantages of a diff-drive robot are its simplicity (typically the motor is attached directly to the axle of each wheel) and high maneuverability (the robot can spin in place by rotating the wheels in opposite directions). Casters are often not appropriate for outdoor use, however.

13.3.1.3 The Car-Like Robot

The most familiar wheeled vehicle is a car, with two steered front wheels and two fixed-heading rear wheels. To prevent slipping of the front wheels, they are steered using **Ackermann steering**, as illustrated in Figure 13.10. The center of rotation of the car's chassis lies on the line passing through the rear wheels at the intersection with the perpendicular bisectors of the front wheels.

To define the configuration of the car, we ignore the rolling angles of the four wheels and write $q = (\phi, x, y, \psi)$, where (x, y) is the location of the midpoint between the rear wheels, ϕ is the car's heading direction, and ψ is the steering angle of the car, defined at a virtual wheel at the midpoint between the front wheels. The controls are the forward speed v of the car at its reference point

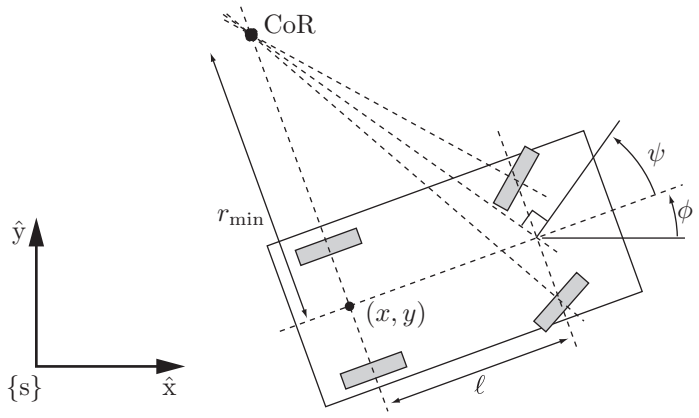

Figure 13.10: The two front wheels of a car are steered at different angles using Ackermann steering such that all wheels roll without slipping (i.e., the wheel heading direction is perpendicular to the line connecting the wheel to the CoR). The car is shown executing a turn at its minimum turning radius $r_{\min}$.

and the angular speed w of the steering angle. The car's kinematics are

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} (\tan\psi)/\ell & 0 \\ \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}, \tag{13.16}$$

where ℓ is the wheelbase between the front and rear wheels. The control v is limited to a closed interval $[v_{\min}, v_{\max}]$ where $v_{\min} < 0 < v_{\max}$, the steering rate is limited to $[-w_{\max}, w_{\max}]$ with $w_{\max} > 0$, and the steering angle ψ is limited to $[-\psi_{\max}, \psi_{\max}]$ with $\psi_{\max} > 0$.

The kinematics (13.16) can be simplified if the steering control is actually just the steering angle ψ and not its rate w. This assumption is justified if the steering rate limit $w_{\max}$ is high enough that the steering angle can be changed nearly instantaneously by a lower-level controller. In this case, ψ is eliminated as a state variable, and the car's configuration is simply $q = (\phi, x, y)$. We use the control inputs (v, ω), where v is still the car's forward speed and ω is now its rate of rotation. These can be converted to the controls (v, ψ) by the relations

$$v = v, \qquad \psi = \tan^{-1}\left(\frac{\ell\omega}{v}\right). \tag{13.17}$$

The constraints on the controls (v, ω) due to the constraints on (v, ψ) take a somewhat complicated form, as we will see shortly.

The simplified car kinematics can now be written

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = G(q)u = \begin{bmatrix} 0 & 1 \\ \cos\phi & 0 \\ \sin\phi & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{13.18}$$

The nonholonomic constraint implied by (13.18) can be derived using one of the equations from (13.18),

$$\dot{x} = v\cos\phi,$$
$$\dot{y} = v\sin\phi,$$

to solve for v, then substituting the result into the other equation to get

$$A(q)\dot{q} = [0 \;\; \sin\phi \;\; -\cos\phi]\dot{q} = \dot{x}\sin\phi - \dot{y}\cos\phi = 0.$$

13.3.1.4 Canonical Simplified Model for Nonholonomic Mobile Robots

The kinematics (13.18) gives a canonical simplified model for nonholonomic mobile robots. Using control transformations such as (13.17), the simplified unicycle kinematics (13.13) and the simplified differential-drive kinematics (13.15) can also be expressed in this form. The control transformation for the simplified unicycle kinematics (13.13) is

$$u_1 = \frac{v}{r}, \qquad u_2 = \omega$$

and the transformation for the simplified diff-drive kinematics (13.15) is

$$u_{\mathrm{L}} = \frac{v - \omega d}{r}, \qquad u_{\mathrm{R}} = \frac{v + \omega d}{r}.$$

With these input transformations, the only difference between the simplified unicycle, diff-drive robot, and car kinematics is the control limits on (v, ω). These are illustrated in Figure 13.11.

We can use the two control inputs (v, ω) in the canonical model (13.18) to directly control the two components of the linear velocity of a reference point P fixed to the robot chassis. This is useful when a sensor is located at P, for example. Let (x_P, y_P) be the coordinates of P in the world frame, and $(x_{\mathrm{r}}, y_{\mathrm{r}})$ be its (constant) coordinates in the chassis frame {b} (Figure 13.12). To find the controls (v, ω) needed to achieve a desired world-frame motion $(\dot{x}_P, \dot{y}_P)$, we first write

$$\begin{bmatrix} x_P \\ y_P \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_{\mathrm{r}} \\ y_{\mathrm{r}} \end{bmatrix}. \tag{13.19}$$

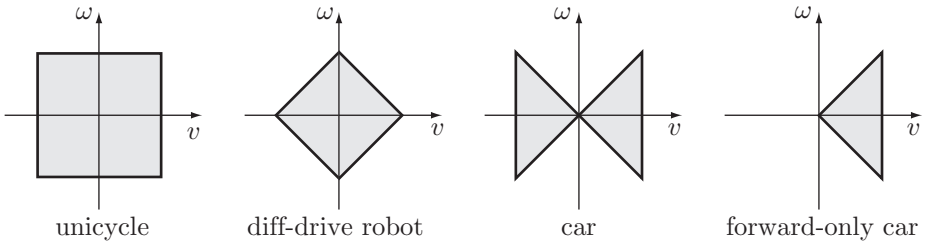

Figure 13.11: The (v, ω) control sets for the simplified unicycle, diff-drive robot, and car kinematics. For the car with a reverse gear, control set illustrates that it is incapable of turning in place. The angle of the sloped lines in its bowtie control set is determined by its minimum turning radius. If a car has no reverse gear, only the right-hand half of the bowtie is available.

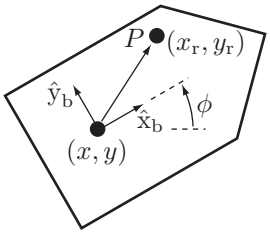

Figure 13.12: The point P is located at (x_r, y_r) in the chassis-fixed frame {b}.

Differentiating, we obtain

$$\begin{bmatrix} \dot{x}_P \\ \dot{y}_P \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} + \dot{\phi} \begin{bmatrix} -\sin\phi & -\cos\phi \\ \cos\phi & -\sin\phi \end{bmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix}. \tag{13.20}$$

Substituting ω for $\dot{\phi}$ and $(v\cos\phi, v\sin\phi)$ for $(\dot{x}, \dot{y})$ and solving, we get

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{1}{x_r} \begin{bmatrix} x_r\cos\phi - y_r\sin\phi & x_r\sin\phi + y_r\cos\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \dot{x}_P \\ \dot{y}_P \end{bmatrix}. \tag{13.21}$$

This equation may be read as $[v\ \omega]^T = J^{-1}(q)[\dot{x}_P\ \dot{y}_P]^T$, where $J(q)$ is the Jacobian relating (v, ω) to the world-frame motion of P. Note that the Jacobian $J(q)$ is singular when P is chosen on the line $x_r = 0$. Points on this line, such as the midway point between the wheels of a diff-drive robot or between the rear wheels of a car, can only move in the heading direction of the vehicle.

13.3.2 Controllability

The feedback control for an omnidirectional robot is simple, as there is a set of wheel driving speeds for any desired chassis velocity $\dot{q}$ (Equation (13.7)). In fact, if the goal of the feedback controller is simply to stabilize the robot to the origin $q = (0, 0, 0)$, rather than trajectory tracking as in the control law (13.11), we could use the even simpler feedback controller

$$\dot{q}(t) = -Kq(t) \tag{13.22}$$

for any positive-definite K. The feedback gain matrix $-K$ acts like a spring to pull q to the origin, and Equation (13.7) is used to transform $\dot{q}(t)$ to $u(t)$. The same type of "linear spring" controller could be used to stabilize the point P on the canonical nonholonomic robot (Figure 13.12) to $(x_P, y_P) = (0, 0)$ since, by Equation (13.21), any desired $(\dot{x}_P, \dot{y}_P)$ can be achieved by the controls (v, ω).[2]

In short, the kinematics of the omnidirectional robot, as well as the kinematics of the point P for the nonholonomic robot, can be rewritten in the single-integrator form

$$\dot{x} = \nu, \tag{13.23}$$

where x is the configuration we are trying to control and ν is a "virtual control" that is actually implemented using the transformations in Equation (13.7) for an omnidirectional robot or Equation (13.21) for the control of P by a nonholonomic robot. Equation (13.23) is a simple example of the more general class of linear control systems

$$\dot{x} = Ax + B\nu, \tag{13.24}$$

which are known to be **linearly controllable** if the **Kalman rank condition** is satisfied:

$$\text{rank}\,[B \;\; AB \;\; A^2B \;\; \cdots \;\; A^{n-1}B] = \dim(x) = n,$$

where $x \in \mathbb{R}^n, \nu \in \mathbb{R}^m, A \in \mathbb{R}^{n\times n}$, and $B \in \mathbb{R}^{n\times m}$. In Equation (13.23), $A = 0$ and B is the identity matrix, trivially satisfying the rank condition for linear controllability since $m = n$. Linear controllability implies the existence of the simple linear control law

$$\nu = -Kx,$$

as in Equation (13.22), to stabilize the origin.

There is no linear controller that can stabilize the full chassis configuration to $q = 0$ for a nonholonomic robot, however; the nonholonomic robot is not linearly controllable. In fact, there is no controller that is a continuous function

[2] For the moment we ignore the different constraints on (v, ω) for the unicycle, diff-drive robot, and car-like robot, as they do not change the main result.

of q which can stabilize $q = 0$. This fact is embedded in the following well-known result, which we state without proof.

Theorem 13.1. *A system $\dot{q} = G(q)u$ with* rank $G(0) <$ dim(q) *cannot be stabilized to $q = 0$ by a continuous time-invariant feedback control law.*

This theorem applies to our canonical nonholonomic robot model, since the rank of $G(q)$ is 2 everywhere (there are only two control vector fields), while the chassis configuration is three dimensional.

For nonlinear systems of the form $\dot{q} = G(q)u$, there are other notions of controllability. We consider a few of these next and show that, even though the canonical nonholonomic robot is not linearly controllable, it still satisfies other important notions of controllability. In particular, the velocity constraint does not integrate to a configuration constraint – the set of reachable configurations is not reduced because of the velocity constraint.

13.3.2.1 Definitions of Controllability

Our definitions of nonlinear controllability rely on the notion of the time- and space-limited reachable sets of the nonholonomic robot from a configuration q.

Definition 13.2. Given a time $T > 0$ and a neighborhood[3] W of an initial configuration q, the **reachable set** of configurations from q at time T by feasible trajectories remaining inside W is written $\mathcal{R}^W(q, T)$. We further define the union of reachable sets at times $t \in [0, T]$:

$$\mathcal{R}^W(q, \leq T) = \bigcup_{0 \leq t \leq T} \mathcal{R}^W(q, t).$$

We now provide some standard definitions of nonlinear controllability.

Definition 13.3. A robot is **controllable** from q if, for any q_{goal}, there exists a control trajectory $u(t)$ that drives the robot from q to q_{goal} in finite time T. The robot is **small-time locally accessible** (STLA) from q if, for any time $T > 0$ and any neighborhood W, the reachable set $\mathcal{R}^W(q, \leq T)$ is a full-dimensional subset of the configuration space. The robot is **small-time locally controllable** (STLC) from q if, for any time $T > 0$ and any neighborhood W, the reachable set $\mathcal{R}^W(q, \leq T)$ is a neighborhood of q.

[3] A neighborhood W of a configuration q is any full-dimensional subset of configuration space containing q in its interior. For example, the set of configurations in a ball of radius $r > 0$ centered at q (i.e., all q_b satisfying $\|q_b - q\| < r$) is a neighborhood of q.

Figure 13.13: Illustrations of small-time local accessibility (STLA) and small-time local controllability (STLC) in a two-dimensional space. The shaded regions are the reachable sets without leaving the neighborhood W.

Small-time local accessibility and small-time local controllability are illustrated in Figure 13.13 for a two-dimensional configuration space. Clearly STLC at q is a stronger condition than STLA at q. If a system is STLC at all q, then it is controllable from any q by the patching together of paths in neighborhoods from q to q_{goal}.

For all the examples in this chapter, if a controllability property holds for any q then it holds for all q, since the maneuverability of the robot does not change with its configuration.

Consider the examples of a car and of a forward-only car with no reverse gear. A forward-only car is STLA, as we will see shortly, but it is not STLC: if it is confined to a tight space (a small neighborhood W), it cannot reach configurations directly behind its initial configuration. A car with a reverse gear is STLC, however. Both cars are controllable in an obstacle-free plane, because even a forward-only car can drive anywhere.

If there are obstacles in the plane, there may be some free-space configurations that the forward-only car cannot reach but that the STLC car can reach. (Consider an obstacle directly in front of the car, for example.) If the obstacles are all defined as closed subsets of the plane containing their boundaries, the STLC car can reach any configuration in its connected component of the free space, despite its velocity constraint.

It is worth thinking about this last statement for a moment. All free configurations have collision-free neighborhoods, since the free space is defined as open and the obstacles are defined as closed (containing their boundaries). Therefore it is always possible to maneuver in any direction from any free configuration. If your car is shorter than the available parking space, you can parallel park into it, even if it takes a long time!

If any controllability property holds (controllability, STLA, or STLC) then the reachable configuration space is full dimensional, and therefore any velocity constraints on the system are nonholonomic.

13.3.2.2 Controllability Tests

Consider a driftless linear-in-the-control (**control-affine**) system

$$\dot{q} = G(q)u = \sum_{i=1}^{m} g_i(q)u_i, \qquad q \in \mathbb{R}^n, \qquad u \in \mathcal{U} \subset \mathbb{R}^m, \quad m < n, \tag{13.25}$$

generalizing the canonical nonholonomic model where $n = 3$ and $m = 2$. The set of feasible controls is $\mathcal{U} \subset \mathbb{R}^m$. For example, the control sets $\mathcal{U}$ for the unicycle, diff-drive, car-like, and forward-only car-like robots were shown in Figure 13.11. In this chapter we consider two types of control sets $\mathcal{U}$: those whose positive linear span is $\mathbb{R}^m$, i.e., $\text{pos}(\mathcal{U}) = \mathbb{R}^m$, such as the control sets for the unicycle, diff-drive robot, and car in Figure 13.11, and those whose positive linear span does not cover $\mathbb{R}^m$ but whose linear span does, i.e., $\text{span}(\mathcal{U}) = \mathbb{R}^m$, such as the control set for the forward-only car in Figure 13.11.

The local controllability properties (STLA or STLC) of (13.25) depend on the noncommutativity of motions along the vector fields g_i. Let $F_\epsilon^{g_i}(q)$ be the configuration reached by following the vector field g_i for time ϵ starting from q. Then two vector fields $g_i(q)$ and $g_j(q)$ commute if $F_\epsilon^{g_j}(F_\epsilon^{g_i}(q)) = F_\epsilon^{g_i}(F_\epsilon^{g_j}(q))$, i.e., the order of following the vector fields does not matter. If they do not commute, i.e., $F_\epsilon^{g_j}(F_\epsilon^{g_i}(q)) - F_\epsilon^{g_i}(F_\epsilon^{g_j}(q)) \neq 0$ then the order of application of the vector fields affects the final configuration. In addition, defining the noncommutativity as

$$\Delta q = F_\epsilon^{g_j}(F_\epsilon^{g_i}(q)) - F_\epsilon^{g_i}(F_\epsilon^{g_j}(q)) \qquad \text{for small } \epsilon,$$

if Δq is in a direction that cannot be achieved directly by any other vector field g_k then switching between g_i and g_j can create motion in a direction not present in the original set of vector fields. A familiar example is parallel parking a car: there is no vector field corresponding to direct sideways translation but, by alternating forward and backward motion along two different vector fields, it is possible to create a net motion to the side.

To calculate $q(2\epsilon) = F_\epsilon^{g_j}(F_\epsilon^{g_i}(q(0)))$ for small ϵ approximately, we use a Taylor expansion and truncate the expansion at $O(\epsilon^3)$. We start by following g_i for a time ϵ and use the fact that $\dot{q} = g_i(q)$ and $\ddot{q} = (\partial g_i/\partial q)\dot{q} = (\partial g_i/\partial q)g_i(q)$:

$$\begin{aligned} q(\epsilon) &= q(0) + \epsilon\dot{q}(0) + \frac{1}{2}\epsilon^2\ddot{q}(0) + O(\epsilon^3) \\ &= q(0) + \epsilon g_i(q(0)) + \frac{1}{2}\epsilon^2 \frac{\partial g_i}{\partial q} g_i(q(0)) + O(\epsilon^3). \end{aligned}$$

Now, after following g_j for a time ϵ:

$$\begin{aligned} q(2\epsilon) &= q(\epsilon) + \epsilon g_j(q(\epsilon)) + \frac{1}{2}\epsilon^2 \frac{\partial g_j}{\partial q} g_j(q(\epsilon)) + O(\epsilon^3) \\ &= q(0) + \epsilon g_i(q(0)) + \frac{1}{2}\epsilon^2 \frac{\partial g_i}{\partial q} g_i(q(0)) \\ &\quad + \epsilon g_j\big(q(0) + \epsilon g_i(q(0))\big) + \frac{1}{2}\epsilon^2 \frac{\partial g_j}{\partial q} g_j(q(0)) + O(\epsilon^3) \\ &= q(0) + \epsilon g_i(q(0)) + \frac{1}{2}\epsilon^2 \frac{\partial g_i}{\partial q} g_i(q(0)) \\ &\quad + \epsilon g_j(q(0)) + \epsilon^2 \frac{\partial g_j}{\partial q} g_i(q(0)) + \frac{1}{2}\epsilon^2 \frac{\partial g_j}{\partial q} g_j(q(0)) + O(\epsilon^3). \end{aligned} \tag{13.26}$$

Note the presence of $\epsilon^2(\partial g_j/\partial q)g_i$, the only term that depends on the order of the vector fields. Using the expression (13.26), we can calculate the noncommutativity:

$$\Delta q = F_\epsilon^{g_j}(F_\epsilon^{g_i}(q)) - F_\epsilon^{g_i}(F_\epsilon^{g_j}(q)) = \epsilon^2 \left(\frac{\partial g_j}{\partial q} g_i - \frac{\partial g_i}{\partial q} g_j \right)(q(0)) + O(\epsilon^3). \tag{13.27}$$

In addition to measuring the noncommutativity, Δq is also equal to the net motion (to order ϵ^2) obtained by following g_i for time ϵ, then g_j for time ϵ, then $-g_i$ for time ϵ, and then $-g_j$ for time ϵ.

The term $(\partial g_j/\partial q)g_i - (\partial g_i/\partial q)g_j$ in Equation (13.27) is important enough for us to give it its own name:

Definition 13.4. The **Lie bracket** of the vector fields $g_i(q)$ and $g_j(q)$ is

$$[g_i, g_j](q) = \left(\frac{\partial g_j}{\partial q} g_i - \frac{\partial g_i}{\partial q} g_j \right)(q). \tag{13.28}$$

This Lie bracket is the same as that for twists, introduced in Section 8.2.2. The only difference is that the Lie bracket in Section 8.2.2 was thought of as the noncommutativity of two twists $\mathcal{V}_i, \mathcal{V}_j$ defined at a given instant, rather than of two velocity vector fields defined over all configurations q. The Lie bracket from Section 8.2.2 would be identical to the expression in Equation (13.28) if the constant twists were represented as vector fields $g_i(q), g_j(q)$ in local coordinates q. See Exercise 13.20, for example.

The Lie bracket of two vector fields $g_i(q)$ and $g_j(q)$ should itself be thought of as a vector field $[g_i, g_j](q)$, where approximate motion along the Lie bracket vector field can be obtained by switching between the original two vector fields.

As we saw in our Taylor expansion, motion along the Lie bracket vector field is slow relative to the motions along the original vector fields; for small times ϵ, motion of order ϵ can be obtained in the directions of the original vector fields, while motion in the Lie bracket direction is only of order ϵ^2. This agrees with our common experience that moving a car sideways by parallel parking motions is slow relative to forward and backward or turning motions, as discussed in the next example.

Example 13.5. Consider the canonical nonholonomic robot with vector fields $g_1(q) = (0, \cos\phi, \sin\phi)$ and $g_2(q) = (1, 0, 0)$. Writing $g_1(q)$ and $g_2(q)$ as column vectors, the Lie bracket vector field $g_3(q) = [g_1, g_2](q)$ is given by

$$\begin{aligned} g_3(q) = [g_1, g_2](q) &= \left(\frac{\partial g_2}{\partial q} g_1 - \frac{\partial g_1}{\partial q} g_2 \right)(q) \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \cos\phi \\ \sin\phi \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ -\sin\phi & 0 & 0 \\ \cos\phi & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ \sin\phi \\ -\cos\phi \end{bmatrix}. \end{aligned}$$

The Lie bracket direction is a sideways "parallel parking" motion, as illustrated in Figure 13.14. The net motion obtained by following g_1 for ϵ, g_2 for ϵ, $-g_1$ for ϵ, and $-g_2$ for ϵ is a motion of order ϵ^2 in this Lie bracket direction, plus a term of order ϵ^3.

From the result of Example 13.5, no matter how small the maneuvering space is for a car with a reverse gear, it can generate sideways motion. Thus we have shown that the Pfaffian velocity constraint implicit in the kinematics $\dot{q} = G(q)u$ for the canonical nonholonomic mobile robot is not integrable to a configuration constraint.

A Lie bracket $[g_i, g_j]$ is called a **Lie product** of degree 2, because the original vector fields appear twice in the bracket. For the canonical nonholonomic model, it is only necessary to consider the degree-2 Lie product to show that there are no configuration constraints. To test whether there are configuration constraints for more general systems of the form (13.25), it may be necessary to consider nested Lie brackets, such as $[g_i, [g_j, g_k]]$ or $[g_i, [g_i, [g_i, g_j]]]$, which are Lie products of degree 3 and 4, respectively. Just as it is possible to generate motions in Lie bracket directions by switching between the original vector fields, it is possible to generate motion in Lie product directions of degree greater than 2. Generating motions in these directions is even slower than for degree-2 Lie products.

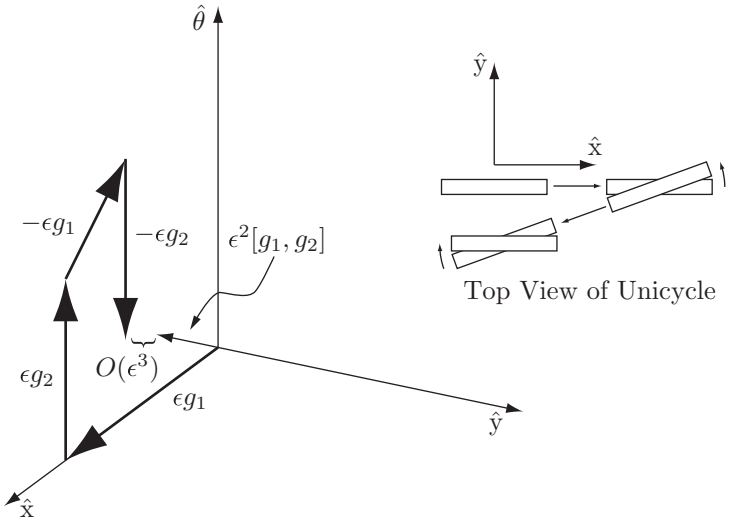

Figure 13.14: The Lie bracket, $[g_1, g_2](q)$, of the forward–backward vector field $g_1(q)$ and the spin-in-place vector field $g_2(q)$ is a sideways vector field.

The **Lie algebra** of a set of vector fields is defined by all Lie products of all degrees, including Lie products of degree 1 (the original vector fields themselves):

Definition 13.6. The **Lie algebra** of a set of vector fields $\mathcal{G} = \{g_1, \ldots, g_m\}$, written $\overline{\mathrm{Lie}}(\mathcal{G})$, is the linear span of all Lie products of degree $1, \ldots, \infty$ of the vector fields $\mathcal{G}$.

For example, for $\mathcal{G} = \{g_1, g_2\}$, $\overline{\mathrm{Lie}}(\mathcal{G})$ is given by the linear combinations of the following Lie products:

degree 1:	g_1, g_2
degree 2:	$[g_1, g_2]$
degree 3:	$[g_1, [g_1, g_2]]; [g_2, [g_1, g_2]]$
degree 4:	$[g_1, [g_1, [g_1, g_2]]]; [g_1, [g_2, [g_1, g_2]]]; [g_2, [g_1, [g_1, g_2]]]; [g_2, [g_2, [g_1, g_2]]]$
$\vdots$	$\vdots$

Since Lie products obey the following identities,

- $[g_i, g_i] = 0$,
- $[g_i, g_j] = -[g_j, g_i]$,

- $[g_i, [g_j, g_k]] + [g_k, [g_i, g_j]] + [g_j, [g_k, g_i]] = 0$ (the Jacobi identity),

not all bracket combinations need to be considered at each degree level.

In practice there will be a finite degree k beyond which higher-degree Lie products yield no more information about the Lie algebra. This happens, for example, when the dimension of the Lie products generated so far is n at all q, i.e., $\dim(\overline{\text{Lie}}(\mathcal{G})(q)) = \dim(q) = n$ for all q; no further Lie brackets can yield new motion directions, as all motion directions have already been obtained. If the dimension of the Lie products generated so far is less than n, however, in general there is no way to know when to stop trying higher-degree Lie products.[4]

With all this as background, we are finally ready to state our main theorem on controllability.

Theorem 13.7. *The control system (13.25), with $\mathcal{G} = \{g_1(q), \ldots, g_m(q)\}$, is small-time locally accessible from q if $\dim(\overline{\text{Lie}}(\mathcal{G})(q)) = \dim(q) = n$ and $\text{span}(\mathcal{U}) = \mathbb{R}^m$. If additionally $\text{pos}(\mathcal{U}) = \mathbb{R}^m$ then the system is small-time locally controllable from q.*

We omit a formal proof, but intuitively we can argue as follows. If the Lie algebra is full rank then the vector fields (followed both forward and backward) locally permit motion in any direction. If $\text{pos}(\mathcal{U}) = \mathbb{R}^m$ (as for a car with a reverse gear) then it is possible to directly follow all vector fields forward or backward, or to switch between feasible controls in order to follow any vector field forward and backward arbitrarily closely, and therefore the Lie algebra rank condition implies STLC. If the controls satisfy only $\text{span}(\mathcal{U}) = \mathbb{R}^m$ (like a forward-only car), then some vector fields may be followed only forward or backward. Nevertheless, the Lie algebra rank condition ensures that there are no equality constraints on the reachable set, so the system is STLA.

For any system of the form (13.25), the question whether the velocity constraints are integrable is finally answered by Theorem 13.7. If the system is STLA at any q, the constraints are not integrable.

Let's apply Theorem 13.7 to a few examples.

Example 13.8 (Controllability of the canonical nonholonomic mobile robot)**.** In Example 13.5 we computed the Lie bracket $g_3 = [g_1, g_2] = (0, \sin\phi, -\cos\phi)$ for the canonical nonholonomic robot. Putting the column vectors $g_1(q)$, $g_2(q)$,

[4] When the system (13.25) is known to be *regular*, however, if there is a degree k that yields no new motion directions not included at lower degrees then there is no need to look at higher-degree Lie products.

and $g_3(q)$ side by side to form a matrix and calculating its determinant, we find

$$\det[\ g_1(q)\ \ g_2(q)\ \ g_3(q)\ ] = \det \begin{bmatrix} 0 & 1 & 0 \\ \cos\phi & 0 & \sin\phi \\ \sin\phi & 0 & -\cos\phi \end{bmatrix} = \cos^2\phi + \sin^2\phi = 1,$$

i.e., the three vector fields are linearly independent at all q, and therefore the dimension of the Lie algebra is 3 at all q. By Theorem 13.7 and the control sets illustrated in Figure 13.11, the unicycle, diff-drive, and car with a reverse gear are STLC at all q, while the forward-only car is only STLA at all q. Each of the unicycle, diff-drive, car, and forward-only car is controllable in an obstacle-free plane.

Example 13.9 (Controllability of the full configuration of the unicycle)**.** We already know from the previous example that the unicycle is STLC on its (ϕ, x, y) subspace; what if we include the rolling angle θ in the description of the configuration? According to Equation (13.12), for $q = (\phi, x, y, \theta)$, the two vector fields are $g_1(q) = (0, r\cos\phi, r\sin\phi, 1)$ and $g_2(q) = (1, 0, 0, 0)$. Calculating the degree-2 and degree-3 Lie brackets

$$g_3(q) = [g_1, g_2](q) = (0, r\sin\phi, -r\cos\phi, 0),$$
$$g_4(q) = [g_2, g_3](q) = (0, r\cos\phi, r\sin\phi, 0),$$

we see that these directions correspond to sideways translation and to forward–backward motion without a change in the wheel rolling angle θ, respectively. These directions are clearly linearly independent of $g_1(q)$ and $g_2(q)$, but we can confirm this by again writing the $g_i(q)$ as column vectors and evaluating

$$\det[\ g_1(q)\ \ g_2(q)\ \ g_3(q)\ \ g_4(q)\ ] = -r^2,$$

i.e., $\dim(\overline{\text{Lie}}(\mathcal{G})(q)) = 4$ for all q. Since $\text{pos}(\mathcal{U}) = \mathbb{R}^2$ for the unicycle, by Figure 13.11, the unicycle is STLC at all points in its four-dimensional configuration space.

You can come to this same conclusion by constructing a short "parallel parking" type maneuver which results in a net change in the rolling angle θ with zero net change in the other configuration variables.

Example 13.10 (Controllability of the full configuration of the diff-drive)**.** The full configuration of the diff-drive is $q = (\phi, x, y, \theta_L, \theta_R)$, including the angles of both wheels. The two control vector fields are given in Equation (13.14). Taking the Lie brackets of these vector fields, we find that we can never create more than four linearly independent vector fields, i.e.,

$$\dim(\overline{\text{Lie}}(\mathcal{G})(q)) = 4$$

at all q. This is because there is a fixed relationship between the two wheel angles (θ_L, θ_R) and the angle of the robot chassis ϕ. Therefore the three velocity constraints $(\dim(q) = 5, \dim(u) = 2)$ implicit in the kinematics (13.14) can be viewed as two nonholonomic constraints and one holonomic constraint. In the full five-dimensional configuration space, the diff-drive is nowhere STLA.

Since usually we worry only about the configuration of the chassis, this negative result is not of much concern.

13.3.3 Motion Planning

13.3.3.1 Obstacle-Free Plane

It is easy to find feasible motions between any two chassis configurations q_0 and q_{goal} in an obstacle-free plane for any of the four nonholonomic robot models (a unicycle, a diff-drive, a car with a reverse gear, and a forward-only car). The problem gets more interesting when we try to optimize an objective function. Below, we consider shortest paths for the forward-only car, shortest paths for the car with a reverse gear, and fastest paths for the diff-drive. The solutions to these problems depend on optimal control theory, and the proofs can be found in the original references (see Section 13.7).

Shortest Paths for the Forward-Only Car The shortest-path problem involves finding a path from q_0 to q_{goal} that minimizes the length of the path that is followed by the robot's reference point. This is not an interesting question for the unicycle or the diff-drive; a shortest path for each of them comprises a rotation to point toward the goal position (x_{goal}, y_{goal}), a translation, and then a rotation to the goal orientation. The total path length is $\sqrt{(x_0 - x_{goal})^2 + (y_0 - y_{goal})^2}$.

The problem is more interesting for the forward-only car, sometimes called the **Dubins car** in honor of the mathematician who first studied the structure of the shortest planar curves with bounded curvature between two oriented points.

Theorem 13.11. *For a forward-only car with the control set shown in Figure 13.11, the shortest paths consist only of arcs at the minimum turning radius and straight-line segments. Denoting a circular arc segment as C and a straight-line segment as S, the shortest path between any two configurations follows either (a) the sequence CSC or (b) the sequence $CC_\alpha C$, where C_α indicates a circular arc of angle $\alpha > \pi$. Any of the C or S segments can be of length zero.*

The two optimal path classes for a forward-only car are illustrated in Figure 13.15. We can calculate the shortest path by enumerating the possible CSC and $CC_\alpha C$ paths. First, construct two minimum-turning-radius circles for the

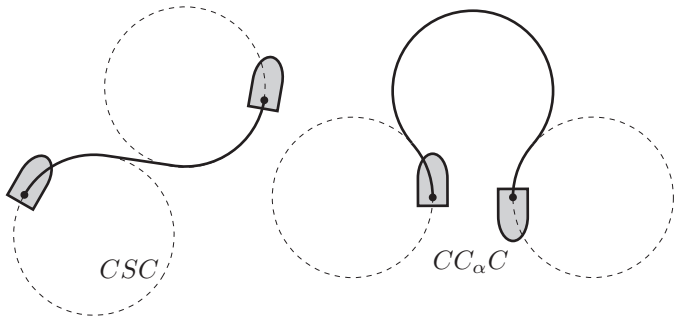

Figure 13.15: The two classes of shortest paths for a forward-only car. The CSC path could be written RSL, and the $CC_\alpha C$ path could be written $LR_\alpha L$.

vehicle at both q_0 and q_{goal} and then solve for (a) the points where lines (with the correct heading direction) are tangent to one of the circles at q_0 and one of the circles at q_{goal}, and (b) the points where a minimum-turning-radius circle (with the correct heading direction) is tangent to one of the circles at q_0 and one of the circles at q_{goal}. The solutions to (a) correspond to CSC paths and the solutions to (b) correspond to $CC_\alpha C$ paths. The shortest of all the solutions is the optimal path. The shortest path may not be unique.

If we break the C segments into two categories, L (when the steering wheel is pegged to the left) and R (when the steering wheel is pegged to the right), we see that there are four types of CSC paths (LSL, LSR, RSL, and RSR) and two types of $CC_\alpha C$ paths ($RL_\alpha R$ and $LR_\alpha L$).

Shortest Paths for the Car with a Reverse Gear The shortest paths for a car with a reverse gear, sometimes called the **Reeds–Shepp car** in honor of the mathematicians who first studied the problem, again use only straight-line segments and minimum-turning-radius arcs. Using the notation C for a minimum-turning-radius arc, C_a for an arc of angle a, S for a straight-line segment, and $|$ for a cusp (a reversal of the linear velocity), Theorem 13.12 enumerates the possible shortest path sequences.

Theorem 13.12. *For a car with a reverse gear with the control set shown in Figure 13.11, the shortest path between any two configurations is in one of the following nine classes:*

$$\begin{array}{ccccc} C|C|C & CC|C & C|CC & CC_a|C_aC & C|C_aC_a|C \\ C|C_{\pi/2}SC & CSC_{\pi/2}|C & C|C_{\pi/2}SC_{\pi/2}|C & CSC & \end{array}$$

Any of the C or S segments can be of length zero.

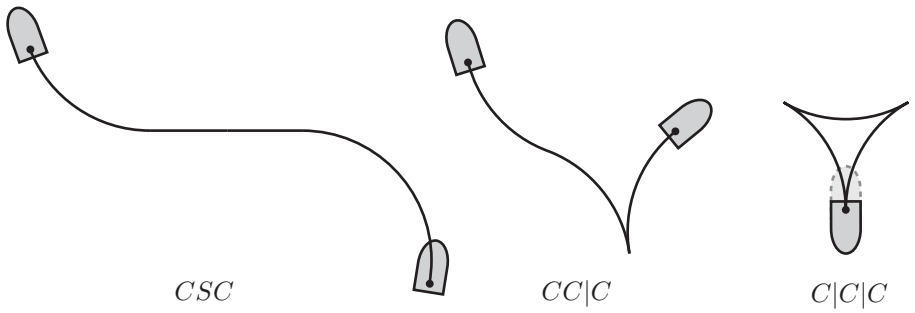

Figure 13.16: Three of the nine classes of shortest paths for a car with a reverse gear.

Three of the nine shortest path classes are illustrated in Figure 13.16. Again, the actual shortest path may be found by enumerating the finite set of possible solutions in the path classes in Theorem 13.12. The shortest path may not be unique.

If we break the C segments into four categories, L^+, L^-, R^+, and R^-, where L and R mean that the steering wheel is turned all the way to the left or right and the superscripts '$+$' and '$-$' indicate the gear shift (forward or reverse), then the nine path classes of Theorem 13.12 can be expressed as $(6\times 4)+(3\times 8)=48$ different types:

6 path classes, each with 4 path types:	$C\|C\|C$, $CC\|C$, $C\|CC$, $CC_a\|C_aC$, $C\|C_aC_a\|C$, $C\|C_{\pi/2}SC_{\pi/2}\|C$
3 path classes, each with 8 path types:	$C\|C_{\pi/2}SC$, $CSC_{\pi/2}\|C$, CSC

The four types for six path classes are determined by the four different initial motion directions, L^+, L^-, R^+, and R^-. The eight types for three path classes are determined by the four initial motion directions and whether the turn is to the left or the right after the straight-line segment. There are only four types in the $C|C_{\pi/2}SC_{\pi/2}|C$ class because the turn after the S segment is always opposite to the turn before the S segment.

If it takes zero time to reverse the linear velocity, a shortest path is also a minimum-time path for the control set for the car with a reverse gear illustrated in Figure 13.11, where the only controls (v,ω) ever used are the two controls $(\pm v_{\max},0)$, an S segment, or the four controls $(\pm v_{\max},\pm\omega_{\max})$, a C segment.

motion segments	number of types	motion sequences
1	4	F, B, R, L
2	8	FR, FL, BR, BL, RF, RB, LF, LB
3	16	FRB, FLB, $FR_\pi B$, $FL_\pi B$, BRF, BLF, $BR_\pi F$, $BL_\pi F$, RFR, RFL, RBR, RBL, LFR, LFL, LBR, LBL
4	8	$FRBL$, $FLBR$, $BRFL$, $BLFR$, $RFLB$, $RBLF$, $LFRB$, $LBRF$
5	4	$FRBLF$, $FLBRF$, $BRFLB$, $BLFRB$

Table 13.1: The 40 time-optimal trajectory types for the diff-drive. The notation R_π and L_π indicate spins of angle π.

Minimum-Time Motions for the Diff-Drive For a diff-drive robot with the diamond-shaped control set in Figure 13.11, any minimum-time motion consists of only translational motions and spins in place.

Theorem 13.13. *For a diff-drive robot with the control set illustrated in Figure 13.11, minimum-time motions consist of forward and backward translations (F and B) at maximum speed $\pm v_{\max}$ and spins in place (R and L for right turns and left turns) at maximum angular speed $\pm\omega_{\max}$. There are 40 types of time-optimal motions, which are categorized in Table 13.1 by the number of motion segments. The notations R_π and L_π indicate spins of angle π.*

Note that Table 13.1 includes both $FR_\pi B$ and $FL_\pi B$, which are equivalent, as well as $BR_\pi F$ and $BL_\pi F$. Each trajectory type is time optimal for some pair $\{q_0, q_{\text{goal}}\}$, and the time-optimal trajectory may not be unique. Notably absent are three-segment sequences where the first and last motions are translations in the same direction (i.e., FRF, FLF, BRB, and BLB).

While any reconfiguration of the diff-drive can be achieved by spinning, translating, and spinning, in some cases other three-segment sequences have a shorter travel time. For example consider a diff-drive with $v_{\max} = \omega_{\max} = 1$, $q_0 = 0$, and $q_{\text{goal}} = (-7\pi/8, 1.924, 0.383)$, as shown in Figure 13.17. The time needed for a spin of angle α is $|\alpha|/\omega_{\max} = |\alpha|$ and the time for a translation of d is $|d|/v_{\max} = |d|$. Therefore, the time needed for the LFR sequence is

$$\frac{\pi}{16} + 1.962 + \frac{15\pi}{16} = 5.103,$$

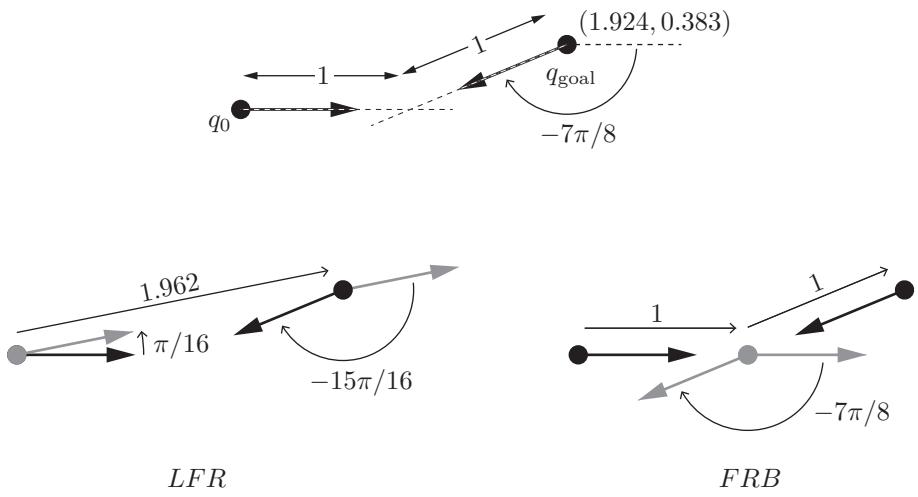

Figure 13.17: (Top) A motion planning problem specified as a motion from $q_0 = (0, 0, 0)$ to $q_{\text{goal}} = (-7\pi/8, 1.924, 0.383)$. (Bottom left) A non-optimal LFR solution taking time 5.103. (Bottom right) The time-optimal FRB solution, through a "via point," taking time 4.749.

while the time needed for the FRB sequence through a "via point" is

$$1 + \frac{7\pi}{8} + 1 = 4.749.$$

13.3.3.2 With Obstacles

If there are obstacles in the plane, the grid-based motion planning methods of Section 10.4.2 can be applied to the unicycle, diff-drive, car with a reverse gear, or forward-only car using discretized versions of the control sets in Figure 13.11. See, for example, the discretizations in Figure 10.14, which use the extremal controls from Figure 13.11. Using extremal controls takes advantage of our observation that shortest paths for reverse-gear cars and the diff-drive consist of minimum-turning-radius turns and straight-line segments. Also, because the C-space is only three dimensional, the grid size should be manageable for reasonable resolutions along each dimension.

We can also apply the sampling methods of Section 10.5. For RRTs, we can again use a discretization of the control set, as mentioned above or, for both

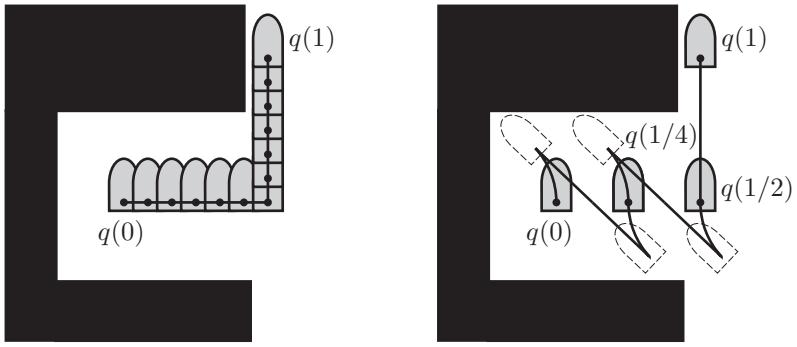

Figure 13.18: (Left) The original path from $q(0)$ to $q(1)$ found by a motion planner that does not respect the reverse-gear car's motion constraints. (Right) The path found by the recursive subdivision has via points at $q(1/4)$ and $q(1/2)$.

PRMs and RRTs, a local planner that attempts to connect two configurations could use the shortest paths from Theorems 13.11, 13.12, or 13.13.

Another option for a reverse-gear car is to use any efficient obstacle-avoiding path planner, even if it ignores the motion constraints of the vehicle. Since such a car is STLC and since the free configuration space is defined to be open (obstacles are closed, containing their boundaries), the car can follow the path found by the planner arbitrarily closely. To follow the path closely, however, the motion may have to be slow – imagine using parallel parking to travel a kilometer down the road.

Alternatively, an initial constraint-free path can be quickly transformed into a fast, feasible, path that respects the car's motion constraints. To do this, represent the initial path as $q(s), s \in [0, 1]$. Then try to connect $q(0)$ to $q(1)$ using a shortest path from Theorem 13.12. If this path is in collision, then divide the original path in half and try to connect $q(0)$ to $q(1/2)$ and $q(1/2)$ to $q(1)$ using shortest paths. If either of these paths are in collision, divide that path, and so on. Because the car is STLC and the initial path lies in open free space, the process will eventually terminate; the new path consists of a sequence of subpaths from Theorem 13.12. The process is illustrated in Figure 13.18.

13.3.4 Feedback Control

We can consider three types of feedback control problems for the canonical nonholonomic mobile robot (13.18) with controls (v, ω):

(a) **Stabilization of a configuration.** Given a desired configuration q_d,

drive the error $q_d - q(t)$ to zero as time goes to infinity. As we saw in Theorem 13.1, no time-invariant feedback law that is continuous in the state variables can stabilize a configuration for a nonholonomic mobile robot. There do exist time-varying and discontinuous feedback laws that accomplish the task, but we do not consider this problem further here.

(b) **Trajectory tracking.** Given a desired trajectory $q_d(t)$, drive the error $q_d(t) - q(t)$ to zero as time goes to infinity.

(c) **Path tracking.** Given a path $q(s)$, follow the geometric path without regard to the time of the motion. This provides more control freedom than the trajectory tracking problem; essentially, we can choose the speed of the reference configuration along the path so as to help reduce the tracking error, in addition to choosing (v, ω).

Path tracking and trajectory tracking are "easier" than stabilizing a configuration, in the sense that there exist continuous time-invariant feedback laws to stabilize the desired motions. In this section we consider the problem of trajectory tracking.

Assume that the reference trajectory is specified as $q_d(t) = (\phi_d(t), x_d(t), y_d(t))$ for $t \in [0, T]$, with a corresponding nominal control $(v_d(t), \omega_d(t)) \in \text{int}(\mathcal{U})$ for $t \in [0, T]$. The requirement that the nominal control be in the interior of the feasible control set $\mathcal{U}$ ensures that some control effort is "left over" to correct small errors. This implies that the reference trajectory is neither a shortest path nor a time-optimal trajectory, since optimal motions saturate the controls. The reference trajectory could be planned using not-quite-extremal controls.

A simple first controller idea is to choose a reference point P on the chassis of the robot (but not on the axis of the two driving wheels), as in Figure 13.12. The desired trajectory $q_d(t)$ is then represented by the desired trajectory of the reference point $(x_{Pd}(t), y_{Pd}(t))$. To track this reference point trajectory, we can use a proportional feedback controller

$$\begin{bmatrix} \dot{x}_P \\ \dot{y}_P \end{bmatrix} = \begin{bmatrix} k_p(x_{Pd} - x_P) \\ k_p(y_{Pd} - y_p) \end{bmatrix}, \tag{13.29}$$

where $k_p > 0$. This simple linear control law is guaranteed to pull the actual position p along with the moving desired position. The velocity $(\dot{x}_P, \dot{y}_P)$ calculated by the control law (13.29) is converted to (v, ω) by Equation (13.21).

The idea is that, as long as the reference point is moving, over time the entire robot chassis will line up with the desired orientation of the chassis. The problem is that the controller may choose the opposite orientation of what is intended; there is nothing in the control law to prevent this. Figure 13.19 shows

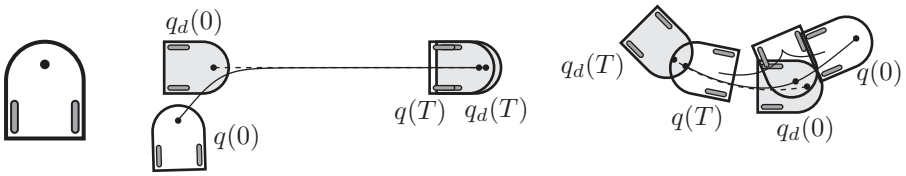

Figure 13.19: (Left) A nonholonomic mobile robot with a reference point. (Middle) A scenario where the linear control law (13.29) tracking a desired reference point trajectory yields the desired trajectory tracking behavior for the entire chassis. (Right) A scenario where the point-tracking control law causes an unintended cusp in the robot motion. The reference point converges to the desired path but the robot's orientation is opposite to the intended orientation.

two simulations, one where the control law (13.29) produces the desired chassis motion and one where the control law causes an unintended reversal in the sign of the driving velocity v. In both simulations the controller succeeds in causing the reference point to track the desired motion.

To fix this, let us explicitly incorporate chassis angle error in the control law. The fixed space frame is {s}, the chassis frame {b} is at the point between the two wheels of the diff-drive (or the two rear wheels for a reverse-gear car) with the forward driving direction along the $\hat{\mathrm{x}}_{\mathrm{b}}$-axis, and the frame corresponding to $q_d(t)$ is {d}. We define the error coordinates

$$q_e = \begin{bmatrix} \phi_e \\ x_e \\ y_e \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_d & \sin\phi_d \\ 0 & -\sin\phi_d & \cos\phi_d \end{bmatrix} \begin{bmatrix} \phi - \phi_d \\ x - x_d \\ y - y_d \end{bmatrix}, \tag{13.30}$$

as illustrated in Figure 13.20. The vector (x_e, y_e) is the {s}-coordinate error vector $(x - x_d, y - y_d)$ expressed in the reference frame {d}.

Consider the nonlinear feedforward plus feedback control law

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} (v_d - k_1|v_d|(x_e + y_e \tan\phi_e))/\cos\phi_e \\ \omega_d - (k_2 v_d y_e + k_3|v_d| \tan\phi_e)\cos^2\phi_e \end{bmatrix}, \tag{13.31}$$

where $k_1, k_2, k_3 > 0$. Note two things about this control law: (1) if the error is zero, the control is simply the nominal control (v_d, ω_d); and (2) the controls grow without bound as ϕ_e approaches $\pi/2$ or $-\pi/2$. In practice, we assume that $|\phi_e|$ is less than $\pi/2$ during trajectory tracking.

In the controller for v, the second term, $-k_1|v_d|x_e/\cos\phi_e$, attempts to reduce x_e by driving the robot so as to catch up with or slow down to the reference frame. The third term, $-k_1|v_d|y_e \tan\phi_e/\cos\phi_e$, attempts to reduce y_e using the component of the forward or backward velocity that impacts y_e.

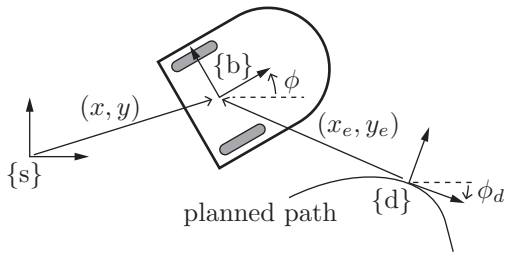

Figure 13.20: The space frame {s}, the robot frame {b}, and the desired configuration {d} driving forward along the planned path. The heading-direction error is $\phi_e = \phi - \phi_d$.

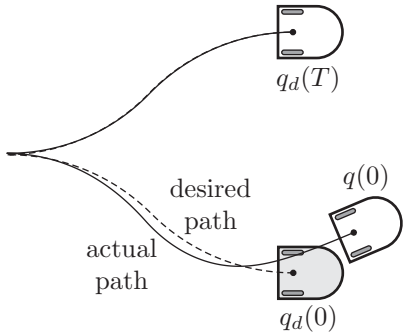

Figure 13.21: A mobile robot implementing the nonlinear control law (13.31).

In the controller for the turning velocity ω, the second term, $-k_2 v_d y_e \cos^2 \phi_e$, attempts to reduce y_e in the future by turning the heading direction of the robot toward the reference-frame origin. The third term, $-k_3 |v_d| \tan \phi_e \cos^2 \phi_e$, attempts to reduce the heading error ϕ_e.

A simulation of the control law (13.31) is shown in Figure 13.21.

The control law requires $|v_d| \neq 0$, so it is not appropriate for stabilizing "spin-in-place" motions for a diff-drive. The proof of the stability of the control law requires methods beyond the scope of this book. In practice, the gains should be chosen large enough to provide significant corrective action but not so large that the controls chatter at the boundary of the feasible control set $\mathcal{U}$.

13.4 Odometry

Odometry is the process of estimating the chassis configuration q from the wheel motions, essentially integrating the effect of the wheel velocities. Since wheel-rotation sensing is available on all mobile robots, odometry is cheap and convenient. Estimation errors tend to accumulate over time, though, due to unexpected slipping and skidding of the wheels and to numerical integration error. Therefore, it is common to supplement odometry with other position sensors, such as GPS, the visual recognition of landmarks, ultrasonic beacons, laser or ultrasonic range sensing, etc. Those sensing modalities have their own measurement uncertainty but errors do not accumulate over time. As a result, odometry generally gives superior results on short time scales, but odometric estimates should either (1) be periodically corrected by other sensing modalities or, preferably, (2) integrated with other sensing modalities in an estimation framework based on a Kalman filter, particle filter, or similar.

In this section we focus on odometry. We assume that each wheel of an omnidirectional robot, and each rear wheel of a diff-drive or car, has an encoder that senses how far the wheel has rotated in its driving direction. If the wheels are driven by stepper motors then we know the driving rotation of each wheel from the steps we have commanded to it.

The goal is to estimate the new chassis configuration q_{k+1} as a function of the previous chassis configuration q_k, given the change in wheel angles from the instant k to the instant $k+1$.

Let $\Delta\theta_i$ be the change in wheel i's driving angle since the wheel angle was last queried a time Δt ago. Since we know only the net change in the wheel driving angle, not the time history of how the wheel angle evolved during the time interval, the simplest assumption is that the wheel's angular velocity was constant during the time interval, $\dot{\theta}_i = \Delta\theta_i/\Delta t$. The choice of units used to measure the time interval is not relevant (since we will eventually integrate the chassis body twist $\mathcal{V}_b$ over the same time interval), so we set $\Delta t = 1$, i.e., $\dot{\theta}_i = \Delta\theta$.

For omnidirectional mobile robots, the vector of wheel speeds $\dot{\theta}$, and therefore $\Delta\theta$, is related to the body twist $\mathcal{V}_b = (\omega_{bz}, v_{bx}, v_{by})$ of the chassis by Equation (13.8):

$$\Delta\theta = H(0)\mathcal{V}_b,$$

where $H(0)$ for the three-omniwheel robot is given by Equation (13.9) and for the four-mecanum-wheel robot is given by Equation (13.10). Therefore, the body twist $\mathcal{V}_b$ corresponding to $\Delta\theta$ is

$$\mathcal{V}_b = H^\dagger(0)\Delta\theta = F\Delta\theta,$$

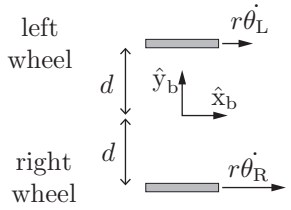

Figure 13.22: The left and right wheels of a diff-drive or the left and right rear wheels of a car.

where $F = H^\dagger(0)$ is the pseudoinverse of $H(0)$. For the three-omniwheel robot,

$$\mathcal{V}_b = F\Delta\theta = r \begin{bmatrix} -1/(3d) & -1/(3d) & -1/(3d) \\ 2/3 & -1/3 & -1/3 \\ 0 & -1/(2\sin(\pi/3)) & 1/(2\sin(\pi/3)) \end{bmatrix} \Delta\theta \quad (13.32)$$

and for the four-mecanum-wheel robot,

$$\mathcal{V}_b = F\Delta\theta = \frac{r}{4} \begin{bmatrix} -1/(\ell+w) & 1/(\ell+w) & 1/(\ell+w) & -1/(\ell+w) \\ 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \Delta\theta. \quad (13.33)$$

The relationship $\mathcal{V}_b = F\dot{\theta} = F\Delta\theta$ also holds for the diff-drive robot and the car (Figure 13.22), where $\Delta\theta = (\Delta\theta_L, \Delta\theta_R)$ (the increments for the left and right wheels) and

$$\mathcal{V}_b = F\Delta\theta = r \begin{bmatrix} -1/(2d) & 1/(2d) \\ 1/2 & 1/2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\theta_L \\ \Delta\theta_R \end{bmatrix}. \quad (13.34)$$

Since the wheel speeds are assumed constant during the time interval, so is the body twist $\mathcal{V}_b$. Calling $\mathcal{V}_{b6}$ the six-dimensional version of the planar twist $\mathcal{V}_b$ (i.e., $\mathcal{V}_{b6} = (0, 0, \omega_{bz}, v_{bx}, v_{by}, 0)$), $\mathcal{V}_{b6}$ can be integrated to generate the displacement created by the wheel-angle increment vector $\Delta\theta$:

$$T_{bb'} = e^{[\mathcal{V}_{b6}]}.$$

From $T_{bb'} \in SE(3)$, which expresses the new chassis frame $\{b'\}$ relative to the initial frame $\{b\}$, we can extract the change in coordinates relative to the body

frame {b}, $\Delta q_b = (\Delta\phi_b, \Delta x_b, \Delta y_b)$, in terms of $(\omega_{bz}, v_{bx}, v_{by})$:

$$\text{if } \omega_{bz} = 0, \quad \Delta q_b = \begin{bmatrix} \Delta\phi_b \\ \Delta x_b \\ \Delta y_b \end{bmatrix} = \begin{bmatrix} 0 \\ v_{bx} \\ v_{by} \end{bmatrix}; \tag{13.35}$$

$$\text{if } \omega_{bz} \neq 0, \quad \Delta q_b = \begin{bmatrix} \Delta\phi_b \\ \Delta x_b \\ \Delta y_b \end{bmatrix} = \begin{bmatrix} \omega_{bz} \\ (v_{bx}\sin\omega_{bz} + v_{by}(\cos\omega_{bz} - 1))/\omega_{bz} \\ (v_{by}\sin\omega_{bz} + v_{bx}(1 - \cos\omega_{bz}))/\omega_{bz} \end{bmatrix}.$$

Transforming Δq_b in {b} to Δq in the fixed frame {s} using the chassis angle ϕ_k,

$$\Delta q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_k & -\sin\phi_k \\ 0 & \sin\phi_k & \cos\phi_k \end{bmatrix} \Delta q_b, \tag{13.36}$$

the updated odometry estimate of the chassis configuration is finally

$$q_{k+1} = q_k + \Delta q.$$

In summary, Δq is calculated using Equations (13.35) and (13.36) as a function of $\mathcal{V}_b$ and the previous chassis angle ϕ_k, and Equation (13.32), (13.33), or (13.34) is used to calculate $\mathcal{V}_b$ as a function of the wheel-angle changes $\Delta\theta$ for the three-omniwheel robot, the four-mecanum-wheel robot, or a nonholonomic robot (the diff-drive or the car), respectively.

13.5 Mobile Manipulation

For a robot arm mounted on a mobile base, **mobile manipulation** describes the coordination of the motion of the base and the robot joints to achieve a desired motion at the end-effector. Typically the motion of the arm can be controlled more precisely than the motion of the base, so the most popular type of mobile manipulation involves driving the base, parking it, letting the arm perform the precise motion task, then driving away.

In some cases, however, it is advantageous, or even necessary, for the end-effector motion to be achieved by a combination of motion of the base and motion of the arm. Defining the fixed space frame {s}, the chassis frame {b}, a frame at the base of the arm {0}, and an end-effector frame {e}, the configuration of {e} in {s} is

$$X(q, \theta) = T_{se}(q, \theta) = T_{sb}(q)\, T_{b0}\, T_{0e}(\theta) \in SE(3),$$

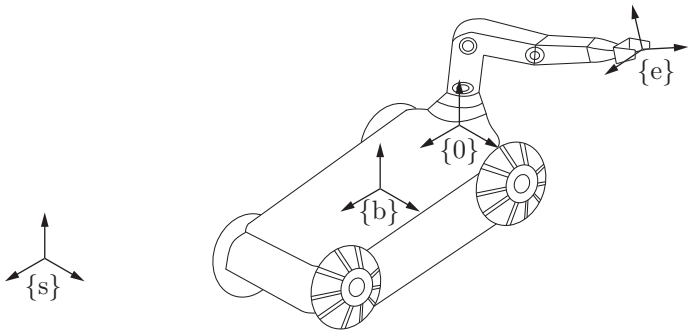

Figure 13.23: The space frame {s} and the frames {b}, {0}, and {e} attached to the mobile manipulator.

where $\theta \in \mathbb{R}^n$ is the set of arm joint positions for the n-joint robot, $T_{0e}(\theta)$ is the forward kinematics of the arm, T_{b0} is the fixed offset of {0} from {b}, $q = (\phi, x, y)$ is the planar configuration of the mobile base, and

$$T_{sb}(q) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & x \\ \sin\phi & \cos\phi & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where z is a constant indicating the height of the {b} frame above the floor. See Figure 13.23.

Let $X(t)$ be the path of the end-effector as a function of time. Then $[\mathcal{V}_e(t)] = X^{-1}(t)\dot{X}(t)$ is the $se(3)$ representation of the end-effector twist expressed in {e}. Further, let the vector of wheel velocities, whether the robot is omnidirectional or nonholonomic, be written $u \in \mathbb{R}^m$. For kinematic control of the end-effector frame using the wheel and joint velocities, we need the Jacobian $J_e(\theta) \in \mathbb{R}^{6\times(m+n)}$ satisfying

$$\mathcal{V}_e = J_e(\theta) \begin{bmatrix} u \\ \dot{\theta} \end{bmatrix} = [J_{\text{base}}(\theta)\ J_{\text{arm}}(\theta)] \begin{bmatrix} u \\ \dot{\theta} \end{bmatrix}.$$

Note that the Jacobian $J_e(\theta)$ does not depend on q: the end-effector velocity expressed in {e} is independent of the configuration of the mobile base. Also, we can partition $J_e(\theta)$ into $J_{\text{base}}(\theta) \in \mathbb{R}^{6\times m}$ and $J_{\text{arm}}(\theta) \in \mathbb{R}^{6\times n}$. The term $J_{\text{base}}(\theta)u$ expresses the contribution of the wheel velocities u to the end-effector's velocity, and the term $J_{\text{arm}}(\theta)\dot{\theta}$ expresses the contribution of the joint velocities to the end-effector's velocity.

In Chapter 5 we developed a method to derive $J_{\text{arm}}(\theta)$, which is called the body Jacobian $J_b(\theta)$ in that chapter. All that remains is to find $J_{\text{base}}(\theta)$. As we saw in Section 13.4, for any type of mobile base there exists an F satisfying

$$\mathcal{V}_b = Fu.$$

To create a six-dimensional twist $\mathcal{V}_{b6}$ corresponding to the planar twist $\mathcal{V}_b$, we can define the $6 \times m$ matrix

$$F_6 = \begin{bmatrix} 0_m \\ 0_m \\ F \\ 0_m \end{bmatrix},$$

where two rows of m zeros are stacked above F and one row is situated below it. Now we have

$$\mathcal{V}_{b6} = F_6 u.$$

This chassis twist can be expressed in the end-effector frame as

$$[\text{Ad}_{T_{eb}(\theta)}]\mathcal{V}_{b6} = [\text{Ad}_{T_{0e}^{-1}(\theta)T_{b0}^{-1}}]\mathcal{V}_{b6} = [\text{Ad}_{T_{0e}^{-1}(\theta)T_{b0}^{-1}}]F_6 u = J_{\text{base}}(\theta)u.$$

Therefore

$$J_{\text{base}}(\theta) = [\text{Ad}_{T_{0e}^{-1}(\theta)T_{b0}^{-1}}]F_6.$$

Now that we have the complete Jacobian $J_e(\theta) = [\, J_{\text{base}}(\theta) \;\; J_{\text{arm}}(\theta)\,]$, we can perform numerical inverse kinematics (Section 6.2) or implement kinematic feedback control laws to track a desired end-effector trajectory. For example, given a desired end-effector trajectory $X_d(t)$, we can choose the kinematic task-space feedforward plus feedback control law (11.16),

$$\mathcal{V}(t) = [\text{Ad}_{X^{-1}X_d}]\mathcal{V}_d(t) + K_p X_{\text{err}}(t) + K_i \int_0^t X_{\text{err}}(\text{t})\, d\text{t}, \tag{13.37}$$

where $[\mathcal{V}_d(t)] = X_d^{-1}(t)\dot{X}_d(t)$, the transform $[\text{Ad}_{X^{-1}X_d}]$ changes the frame of representation of the feedforward twist $\mathcal{V}_d$ from the frame at X_d to the actual end-effector frame at X, and $[X_{\text{err}}] = \log(X^{-1}X_d)$. The commanded end-effector-frame twist $\mathcal{V}(t)$ is implemented as

$$\begin{bmatrix} u \\ \dot{\theta} \end{bmatrix} = J_e^{\dagger}(\theta)\mathcal{V}.$$

As discussed in Section 6.3, it is possible to use a weighted pseudoinverse to penalize certain wheel or joint velocities.

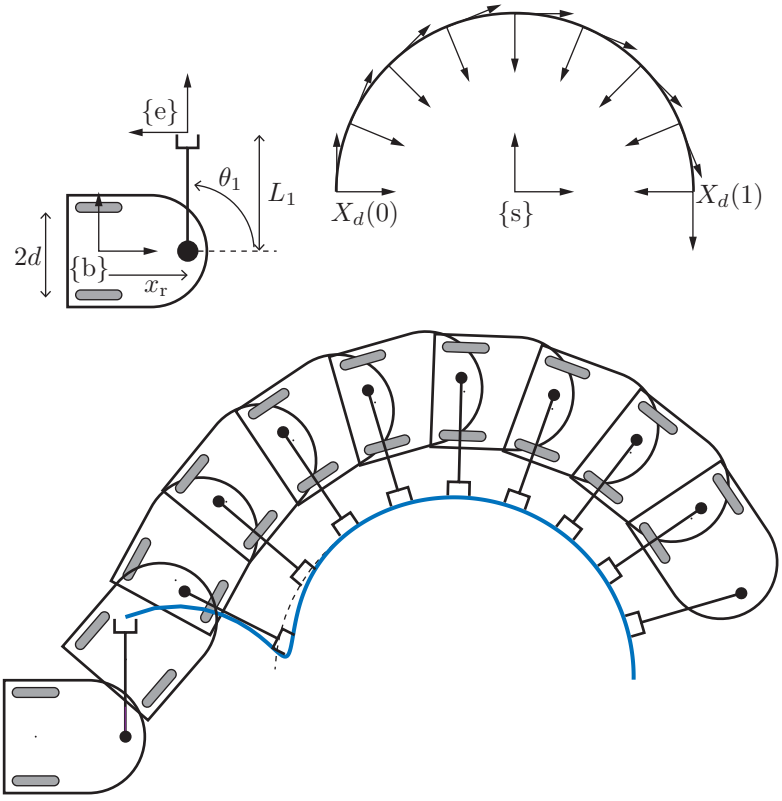

Figure 13.24: A diff-drive with a 1R planar arm with end-effector frame {e}. (Top) The initial configuration of the robot and the desired end-effector trajectory $X_d(t)$. (Bottom) Trajectory tracking using the control law (13.37). The end-effector shoots past the desired path before settling into accurate trajectory tracking.

An example is shown in Figure 13.24. The mobile base is a diff-drive and the arm moves in the plane with only one revolute joint. The desired motion of the end-effector $X_d(t), t \in [0, 1]$, is parametrized by $\alpha = -\pi t$, $x_d(t) = -3\cos(\pi t)$, and $y_d(t) = 3\sin(\pi t)$, where α indicates the planar angle from the $\hat{\mathrm{x}}_\mathrm{s}$-axis to the $\hat{\mathrm{x}}_\mathrm{e}$-axis (see Figure 13.24). The performance in Figure 13.24 demonstrates a bit of overshoot, indicating that the diagonal gain matrix $K_i = k_i I$ should use somewhat lower gains. Alternatively, the gain matrix $K_p = k_p I$ could use larger gains, provided that there are no practical problems with these larger gains (see

the discussion in Section 11.3.1.2).

Note that, for an arbitrary $X_d(t)$ to be feasible for the mobile manipulator, the Jacobian $J_e(\theta)$ should be full rank everywhere; see Exercise 13.30.

13.6 Summary

- The chassis configuration of a wheeled mobile robot moving in the plane is $q = (\phi, x, y)$. The velocity can be represented either as $\dot{q}$ or as the planar twist $\mathcal{V}_b = (\omega_{bz}, v_{bx}, v_{by})$ expressed in the chassis-fixed frame {b}, where

$$\mathcal{V}_b = \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix}.$$

- The chassis of a nonholonomic mobile robot is subject to a single nonintegrable Pfaffian velocity constraint $A(q)\dot{q} = [0 \;\; \sin\phi \;\; -\cos\phi]\dot{q} = \dot{x}\sin\phi - \dot{y}\cos\phi = 0$. An omnidirectional robot, employing omniwheels or mecanum wheels, has no such constraint.

- For a properly constructed omnidirectional robot with $m \geq 3$ wheels, there exists a rank 3 matrix $H(\phi) \in \mathbb{R}^{m\times 3}$ that maps the chassis velocity $\dot{q}$ to the wheel driving velocities u:

$$u = H(\phi)\dot{q}.$$

  In terms of the body twist $\mathcal{V}_b$,

$$u = H(0)\mathcal{V}_b.$$

  The driving speed limits of each wheel place two parallel planar constraints on the feasible body twists, creating a polyhedron V of feasible body twists.

- Motion planning and feedback control for omnidirectional robots is simplified by the fact that there are no chassis velocity equality constraints.

- Nonholonomic mobile robots are described as driftless linear-in-the-control systems

$$\dot{q} = G(q)u, \qquad u \in \mathcal{U} \subset \mathbb{R}^m,$$

  where $G(q) \in \mathbb{R}^{n\times m}, n > m$. The m columns $g_i(q)$ of $G(q)$ are called the control vector fields.

- The canonical simplified nonholonomic mobile robot model is

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = G(q)u = \begin{bmatrix} 0 & 1 \\ \cos\phi & 0 \\ \sin\phi & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}.$$

  The control sets $\mathcal{U}$ differ for the unicycle, diff-drive, reverse-gear car, and forward-only car.

- A control system is small-time locally accessible (STLA) from q if, for any time $T > 0$ and any neighborhood W, the reachable set in time less than T without leaving W is a full-dimensional subset of the configuration space. A control system is small-time locally controllable (STLC) from q if, for any time $T > 0$ and any neighborhood W, the reachable set in time less than T without leaving W is a neighborhood of q. If the system is STLC from a given q, it can maneuver locally in any direction.

- The Lie bracket of two vector fields g_1 and g_2 is the vector field

$$[g_1, g_2] = \left(\frac{\partial g_2}{\partial q} g_1 - \frac{\partial g_1}{\partial q} g_2 \right).$$

- A Lie product of degree k is a Lie bracket term where the original vector fields appear a total of k times. A Lie product of degree 1 is just one of the original vector fields.

- The Lie algebra of a set of vector fields $\mathcal{G} = \{g_1, \ldots, g_m\}$, written $\overline{\text{Lie}}(\mathcal{G})$, is the linear span of all Lie products of degree $1, \ldots, \infty$ of the vector fields $\mathcal{G}$.

- A driftless control-affine system is small-time locally accessible from q if $\dim(\overline{\text{Lie}}(\mathcal{G})(q)) = \dim(q) = n$ and $\text{span}(\mathcal{U}) = \mathbb{R}^m$. If additionally $\text{pos}(\mathcal{U}) = \mathbb{R}^m$ then the system is small-time locally controllable from q.

- For a forward-only car in an obstacle-free plane, shortest paths always follow a turn at the tightest turning radius (C) or straight-line motions (S). There are two classes of shortest paths: CSC and $CC_\alpha C$, where C_α is a turn of angle $|\alpha| > \pi$. Any C or S segment can be of length zero.

- For a car with a reverse gear, shortest paths always consist of a sequence of straight-line segments or turns at the tightest turning radius. Shortest paths always belong to one of nine classes.

- For the diff-drive, minimum-time motions always consist of turn-in-place motions and straight-line motions.
- For the canonical nonholonomic robot, there is no time-invariant control law which is continuous in the configuration and which will stabilize the origin configuration. Continuous time-invariant control laws exist that will stabilize a trajectory, however.
- Odometry is the process of estimating the chassis configuration on the basis of how far the robot's wheels have rotated in their driving direction, assuming no skidding in the driving direction and, for typical wheels (not omniwheels or mecanum wheels), no slip in the orthogonal direction.
- For a mobile manipulator with m wheels and n joints in the robot arm, the end-effector twist $\mathcal{V}_e$ in the end-effector frame $\{e\}$ is written

$$\mathcal{V}_e = J_e(\theta) \begin{bmatrix} u \\ \dot{\theta} \end{bmatrix} = [J_{\text{base}}(\theta)\ J_{\text{arm}}(\theta)] \begin{bmatrix} u \\ \dot{\theta} \end{bmatrix}.$$

  The $6 \times m$ Jacobian $J_{\text{base}}(\theta)$ maps the wheel velocities u to a velocity at the end-effector, and the $6 \times n$ Jacobian $J_{\text{arm}}(\theta)$ is the body Jacobian derived in Chapter 5. The Jacobian $J_{\text{base}}(\theta)$ is given by

$$J_{\text{base}}(\theta) = [\text{Ad}_{T_{0e}^{-1}(\theta) T_{b0}^{-1}}] F_6$$

  where F_6 is the transformation from the wheel velocities to the chassis twist, $\mathcal{V}_{b6} = Fu$.

13.7 Notes and References

Excellent references on modeling, motion planning, and control of mobile robots include the books [33, 81], the chapters in the textbook [171] and the Handbook of Robotics [121, 157], and the encyclopedia chapter [128].

General references on nonholonomic systems, underactuated systems, and notions of nonlinear controllability include [14, 21, 27, 63, 64, 122, 126, 158], the Control Handbook chapter [101], and the encyclopedia chapter [100]. Theorem 13.1 is a strengthening of a result originally reported by Brockett in [19]. Theorem 13.7 is an application of Chow's theorem [28] considering different possible control sets. A more general condition describing the conditions under which Chow's theorem can be used to determine local controllability was given by Sussmann [182].

The original results for the shortest paths for a forward-only car and for a car with a reverse gear were given by Dubins [41] and Reeds and Shepp [146], respectively. These results were extended and applied to motion planning problems in [16, 175] and independently derived using principles of differential geometry in [183]. The minimum-time motions for a diff-drive were derived by Balkcom and Mason in [5]. The motion planner for a car-like mobile robot based on replacing segments of an arbitrary path with shortest feasible paths for a car, described in Section 13.3.3.2, was described in [82].

The nonlinear control law (13.31) for tracking a reference trajectory for a nonholonomic mobile robot is taken from [121, 157].

13.8 Exercises

Exercise 13.1 In the omnidirectional mobile robot kinematic modeling of Section 13.2.1, we derived the relationship between wheel velocities and chassis velocity in what seemed to be an unusual way. First, we specified the chassis velocity, then we calculated how the wheels must be driving (and sliding). At first glance, this approach does not seem to make sense causally; we should specify the velocities of the wheels, then calculate the chassis velocity. Explain mathematically why this modeling approach makes sense, and under what condition the method cannot be used.

Exercise 13.2 According to the kinematic modeling of Section 13.2.1, each wheel of an omnidirectional robot adds two more velocity constraints on the chassis twist $\mathcal{V}_b$. This might seem counterintuitive, since more wheels means more motors and we might think that having more motors should result in more motion capability, not more constraints. Explain clearly why having extra wheels implies extra velocity constraints, in our kinematic modeling, and which assumptions in the kinematic modeling may be unrealistic.

Exercise 13.3 For the three-omniwheel robot of Figure 13.5, is it possible to drive the wheels so that they skid? (In other words, so that the wheels slip in the driving direction.) If so, give an example set of wheel velocities.

Exercise 13.4 For the four-mecanum-wheel robot of Figure 13.5, is it possible to drive the wheels so that they skid? (In other words, the wheels slip in the

driving direction.) If so, give an example set of wheel velocities.

Exercise 13.5 Replace the wheels of the four-mecanum-wheel robot of Figure 13.5 by wheels with $\gamma = \pm 60^\circ$. Derive the matrix $H(0)$ in the relationship $u = H(0)\mathcal{V}_b$. Is it rank 3? If necessary, you can assume values for ℓ and w.

Exercise 13.6 Consider the three-omniwheel robot of Figure 13.5. If we replace the omniwheels by mecanum wheels with $\gamma = 45^\circ$, is it still a properly constructed omnidirectional mobile robot? In other words, in the relationship $u = H(0)\mathcal{V}_b$, is $H(0)$ rank 3?

Exercise 13.7 Consider a mobile robot with three mecanum wheels for which $\gamma = \pm 45^\circ$ at the points of an equilateral triangle. The chassis frame {b} is at the center of the triangle. The driving directions of all three wheels are the same (e.g., along the body $\hat{\mathrm{x}}_\mathrm{b}$-axis) and the free sliding directions are $\gamma = 45^\circ$ for two wheels and $\gamma = -45^\circ$ for the other wheel. Is this a properly constructed omnidirectional mobile robot? In other words, in the relationship $u = H(0)\mathcal{V}_b$, is $H(0)$ rank 3?

Exercise 13.8 Using your favorite graphics software (e.g., MATLAB), plot the two planes bounding the set of feasible body twists $\mathcal{V}_b$ for wheel 2 of the three-omniwheel robot of Figure 13.5.

Exercise 13.9 Using your favorite graphics software (e.g., MATLAB), plot the two planes bounding the set of feasible body twists $\mathcal{V}_b$ for wheel 1 of the four-mecanum-wheel robot of Figure 13.5.

Exercise 13.10 Consider a four-omniwheel mobile robot with wheels at the points of a square. The chassis frame {b} is at the center of the square, and the driving direction of each wheel is in the direction 90° counterclockwise from the vector from the the origin of {b} to the wheel. You may assume that the sides of the squares have length 2. Find the matrix $H(0)$. Is it rank 3?

Exercise 13.11 Implement a collision-free grid-based planner for an omnidirectional robot. You may assume that the robot has a circular chassis, so for collision-detection purposes, you need to consider only the (x, y) location of the robot. The obstacles are circles with random center locations and random radii. You can use Dijkstra's algorithm or A^* to find a shortest path that avoids

obstacles.

Exercise 13.12 Implement an RRT planner for an omnidirectional robot. As above, you may assume a circular chassis and circular obstacles.

Exercise 13.13 Implement a feedforward plus proportional feedback controller to track a desired trajectory for an omnidirectional mobile robot. Test it on the desired trajectory $(\phi_d(t), x_d(t), y_d(t)) = (t, 0, t)$ for $t \in [0, \pi]$. The initial configuration of the robot is $q(0) = (-\pi/4, 0.5, -0.5)$. Plot the configuration error as a function of time. You can also show an animation of the robot converging to the trajectory.

Exercise 13.14 Write down the Pfaffian constraints $A(q)\dot{q} = 0$ corresponding to the unicycle model in Equation (13.12).

Exercise 13.15 Write down the Pfaffian constraints corresponding to the diff-drive model in Equation (13.14).

Exercise 13.16 Write down the Pfaffian constraints corresponding to the car-like model in Equation (13.16).

Exercise 13.17 Give examples of two systems that are STLA but not STLC. The systems should not be wheeled mobile robots.

Exercise 13.18 Continue the Taylor expansion in Equation (13.26) to find the net motion (to order ϵ^2) obtained by following g_i for time ϵ, then g_j for time ϵ, then $-g_i$ for time ϵ, then $-g_j$ for time ϵ. Show that it is equivalent to the expression (13.27).

Exercise 13.19 Write down the canonical nonholonomic mobile robot model (13.18) in the chassis-fixed form

$$\mathcal{V}_b = B \begin{bmatrix} v \\ \omega \end{bmatrix},$$

where B is a 3×2 matrix whose columns correspond to the chassis twist associated with the controls v and ω.

Exercise 13.20 Throughout this book, we have been using the configuration spaces $SE(3)$ and $SO(3)$, and their planar subsets $SE(2)$ and $SO(2)$. These are

known as matrix Lie groups. Body and spatial twists are represented in matrix form as elements of $se(3)$ (or $se(2)$ in the plane) and body and spatial angular velocities are represented in matrix form as elements of $so(3)$ (or $so(2)$ in the plane). The spaces $se(3)$ and $so(3)$ correspond to all possible $\dot{T}$ and $\dot{R}$ when T or R is the identity matrix. Since these spaces correspond to all possible velocities, each is called the Lie algebra of its respective matrix Lie group. Let us call G a matrix Lie group and $\mathfrak{g}$ its Lie algebra, and let X be an element of G, and A and B be elements of $\mathfrak{g}$. In other words, A and B can be thought of as possible values of $\dot{X}$ when $X = I$.

Any "velocity" A in $\mathfrak{g}$ can be "translated" to a velocity $\dot{X}$ at any $X \in G$ by pre-multiplying or post-multiplying by X, i.e., $\dot{X} = XA$ or $\dot{X} = AX$. If we choose $\dot{X} = XA$, i.e., $A = X^{-1}\dot{X}$ then we can think of A as a "body velocity" (e.g., the matrix form of a body twist if $G = SE(3)$) and if we choose $\dot{X} = AX$, i.e., $A = \dot{X}X^{-1}$, we can think of A as a "spatial velocity" (e.g., the matrix form of a spatial twist if $G = SE(3)$). In this way, A can be extended to an entire vector field over G. If the extension is obtained by multiplying by X on the left then the vector field is called left-invariant (constant in the body frame), and if the extension is by multiplying by X on the right then the vector field is called right-invariant (constant in the space frame). Velocities that are constant in the body frame, like the vector fields for the canonical nonholonomic mobile robot, correspond to left-invariant vector fields.

Just as we can define a Lie bracket of two vector fields, as in Equation (13.28), we can define the Lie bracket of $A, B \in \mathfrak{g}$ as

$$[A, B] = AB - BA, \tag{13.38}$$

as described in Section 8.2.2 for $\mathfrak{g} = se(3)$. Confirm that this formula describes the same Lie bracket vector field as Equation (13.28) for the canonical nonholonomic vector fields $g_1(q) = (0, \cos\phi, \sin\phi)$ and $g_2(q) = (1, 0, 0)$. To do this, first express the two vector fields as $A_1, A_2 \in se(2)$, considered to be the generators of the left-invariant vector fields $g_1(q)$ and $g_2(q)$ (since the vector fields correspond to constant velocities in the chassis frame). Then take the Lie bracket $A_3 = [A_1, A_2]$ and extend A_3 to a left-invariant vector field defined at all $X \in SE(2)$. Show that this is the same result obtained using the formula (13.28).

The Lie bracket formula in Equation (13.28) is general for any vector fields expressed as a function of coordinates q, while the formula (13.38) is particularly for left- and right-invariant vector fields defined by elements of the Lie algebra of a matrix Lie group.

Exercise 13.21 Using your favorite symbolic math software (e.g., Mathe-

matica), write or experiment with software that symbolically calculates the Lie bracket of two vector fields. Show that it correctly calculates Lie brackets for vector fields of any dimension.

Exercise 13.22 For the full five-dimensional diff-drive model described by Equation (13.14), calculate Lie products that generate two motion directions not present in the original two vector fields. Write down the holonomic constraint corresponding to the direction in which it is not possible to generate motions.

Exercise 13.23 Implement a collision-free grid-based motion planner for a car-like robot among obstacles using techniques from Section 10.4.2. Decide how to specify the obstacles.

Exercise 13.24 Implement a collision-free RRT-based motion planner for a car-like robot among obstacles using techniques from Section 10.5. Decide how to specify obstacles.

Exercise 13.25 Implement the reference point trajectory-tracking control law (13.29) for a diff-drive robot. Show in a simulation that it succeeds in tracking a desired trajectory for the point.

Exercise 13.26 Implement the nonlinear feedforward plus feedback control law (13.31). Demonstrate its performance in tracking a reference trajectory with different sets of control gains, including one set that yields "good" performance.

Exercise 13.27 Write a program that accepts a time history of wheel encoder values for the two rear wheels of a car and estimates the chassis configuration as a function of time using odometry. Prove that it yields correct results for a chassis motion that involves rotations and translations.

Exercise 13.28 Write a program that accepts a time history of wheel encoder values for the three wheels of a three-omniwheel robot and estimates the chassis configuration as a function of time using odometry. Prove that it yields correct results for a chassis motion that involves rotations and translations.

Exercise 13.29 Write a program that accepts a time history of wheel encoder values for the four wheels of a four-mecanum-wheel robot and estimates the chassis configuration as a function of time using odometry. Prove that it yields

correct results for a chassis motion that involves rotations and translations.

Exercise 13.30 Consider the mobile manipulator in Figure 13.24. Write down the Jacobian $J_e(\theta)$ as a 3×3 matrix function of d, x_r, L_1, and θ_1. Is the Jacobian rank 3 for all choices of d, x_r, L_1, and θ_1? If not, under what conditions is it not full rank?

Exercise 13.31 Write a simulation for a mobile manipulation controller similar to that demonstrated in Figure 13.24. For this simulation you need to include a simulation of odometry to keep track of the mobile base's configuration. Demonstrate your controller on the same example trajectory and initial conditions shown in Figure 13.24 for good and bad choices of control gains.

Exercise 13.32 Wheel-based odometry can be supplemented by odometry based on an inertial measurement unit (IMU). A typical IMU includes a three-axis gyro, for sensing angular velocities of the chassis, and a three-axis accelerometer, for sensing linear accelerations of the chassis. From a known initial state of the mobile robot (e.g., at rest at a known position), the sensor data from the IMU can be integrated over time to yield a position estimate of the robot. Since the data are numerically integrated once in the case of angular velocities, and twice in the case of linear accelerations, the estimate will drift from the actual value over time, just as the wheel-based odometry estimate will.

In a paragraph describe operating conditions for the mobile robot, including properties of the wheels' interactions with the ground, where IMU-based odometry might be expected to yield better configuration estimates, and conditions where wheel-based odometry might be expected to yield better estimates. In another paragraph describe how the two methods can be used simultaneously, to improve the performance beyond that of either method alone. You should feel free to do an internet search and comment on specific data-fusion tools or filtering techniques that might be useful.

Exercise 13.33 The KUKA youBot (Figure 13.25) is a mobile manipulator consisting of a 5R arm mounted on an omnidirectional mobile base with four mecanum wheels. The chassis frame {b} is centered between the four wheels at a height $z = 0.0963$ m above the floor, and the configuration of the chassis

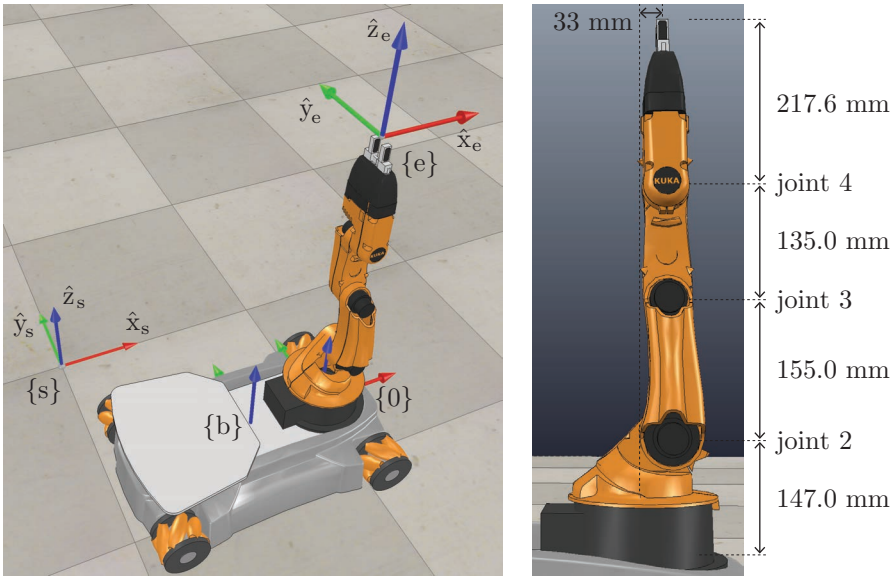

Figure 13.25: (Left) The KUKA youBot mobile manipulator and the fixed space frame {s}, the chassis frame {b}, the arm base frame {0}, and the end-effector frame {e}. The arm is at its zero configuration. (Right) A close-up of the arm at its zero configuration. Joint axes 1 and 5 (not shown) point upward and joint axes 2, 3, and 4 are out of the page.

relative to a fixed space frame {s} is

$$T_{sb}(q) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & x \\ \sin\phi & \cos\phi & 0 & y \\ 0 & 0 & 1 & 0.0963 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where $q = (\phi, x, y)$. The kinematics of the four-wheeled mobile base is described in Figure 13.5 and the surrounding text, where the front–back distance between the wheels is $2\ell = 0.47$ m, the side-to-side distance between the wheels is $2w = 0.3$ m, and the radius of each wheel is $r = 0.0475$ m.

The fixed offset from the chassis frame {b} to the base frame of the arm {0} is

$$T_{b0} = \begin{bmatrix} 1 & 0 & 0 & 0.1662 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.0026 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

i.e., the arm base frame $\{0\}$ is aligned with the chassis frame $\{b\}$ and is displaced by 166.2 mm in $\hat{x}_b$ and 2.6 mm in $\hat{z}_b$. The end-effector frame $\{e\}$ at the zero configuration of the arm (as shown in Figure 13.25) relative to the base frame $\{0\}$ is

$$M_{0e} = \begin{bmatrix} 1 & 0 & 0 & 0.0330 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.6546 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

You can ignore the arm joint limits in this exercise.

(a) Examining the right-hand side of Figure 13.25 – and keeping in mind that (i) joint axes 1 and 5 point up on the page and joint axes 2, 3, and 4 point out of the page, and (ii) positive rotation about an axis is by the right-hand rule – either confirm that the screw axes in the end-effector frame $\mathcal{B}_i$ are as shown in the following table:

i	ω_i	v_i
1	$(0, 0, 1)$	$(0, 0.0330, 0)$
2	$(0, -1, 0)$	$(-0.5076, 0, 0)$
3	$(0, -1, 0)$	$(-0.3526, 0, 0)$
4	$(0, -1, 0)$	$(-0.2176, 0, 0)$
5	$(0, 0, 1)$	$(0, 0, 0)$

or provide the correct $\mathcal{B}_i$.

(b) The robot arm has only five joints, so it is incapable of generating an arbitrary end-effector twist $\mathcal{V}_e \in \mathbb{R}^6$ when the mobile base is parked. If we are able to move the mobile base and the arm joints simultaneously, are there configurations θ of the arm at which arbitrary twists are not possible? If so, indicate these configurations. Also explain why the configuration $q = (\phi, x, y)$ of the mobile base is irrelevant to this question.

(c) Use numerical inverse kinematics to find a chassis and arm configuration (q, θ) that places the end-effector at

$$X(q, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1.0 \\ 0 & -1 & 0 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

You can try $q_0 = (\phi_0, x_0, y_0) = (0, 0, 0)$ and $\theta_0 = (0, 0, -\pi/2, 0, 0)$ as your initial guess.

(d) You will write a robot simulator to test the kinematic task-space feedforward plus feedback control law (13.37) tracking the end-effector trajectory

defined by the path

$$X_d(s) = \begin{bmatrix} \sin(s\pi/2) & 0 & \cos(s\pi/2) & s \\ 0 & 1 & 0 & 0 \\ -\cos(s\pi/2) & 0 & \sin(s\pi/2) & 0.491 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad s \in [0, 1],$$

and the time scaling

$$s(t) = \frac{3}{25}t^2 - \frac{2}{125}t^3, \qquad t \in [0, 5].$$

In other words, the total time of the motion is 5 s.
Your program should take the trajectory and initial configuration of the robot as input, in addition to the control gains and any other parameters you see fit. In this exercise, due to initial error, the initial configuration of the robot is not on the path: $q_0 = (\phi_0, x_0, y_0) = (-\pi/8, -0.5, 0.5)$ and $\theta_0 = (0, -\pi/4, \pi/4, -\pi/2, 0)$.
Your main program loop should run 100 times per simulated second, i.e., each time step is $\Delta t = 0.01$ s, for a total of 500 time steps for the simulation. Each time through the loop, your program should:

- Calculate the desired configuration X_d and twist $\mathcal{V}_d$ at the current time.
- Calculate the current configuration error $X_{\text{err}} = (\omega_{\text{err}}, v_{\text{err}})$ and save X_{err} in an array for later plotting.
- Evaluate the control law (13.37) to find the commanded wheel speeds and joint velocities $(u, \dot{\theta})$.
- Step the simulation of the robot's motion forward in time by Δt to find the new configuration of the robot. You may use a simple first-order Euler integration for the arm: the new joint angle is just the old angle plus the commanded joint velocity multiplied by Δt. To calculate the new configuration of the chassis, use odometry from Section 13.4, remembering that the change in wheel angles during one simulation step is $u\Delta t$.

You are encouraged to test the feedforward portion of your controller first. A starting configuration on the path is approximately $q_0 = (\phi_0, x_0, y_0) = (0, -0.526, 0)$ and $\theta_0 = (0, -\pi/4, \pi/4, -\pi/2, 0)$. Once you have confirmed that your feedforward controller works as expected, add a nonzero proportional gain to get good performance from the configuration with initial error. Finally, you can add a nonzero integral gain to see transient effects such as overshoot.

After the simulation completes, plot the six components of X_{err} as a function of time. If possible, choose gains K_p and K_i such that it is possible to see typical features of a PI velocity controller: a little bit of overshoot and oscillation and eventually nearly zero error. You should choose control gains such that the 2% settling time is one or two seconds, so the transient response is clearly visible. If you have the visualization tools available, create a movie of the robot's motion corresponding to your plots.

(e) While retaining stability, choose a set of different control gains that gives a visibly different behavior of the robot. Provide the plots and movie and comment on why the different behavior agrees, or does not agree, with what you know about PI velocity control.

Exercise 13.34 One type of wheeled mobile robot, not considered in this chapter, has three or more conventional wheels which are all individually steerable. Steerable conventional wheels allow the robot chassis to follow arbitrary paths without relying on the passive sideways rolling of mecanum wheels or omniwheels.

In this exercise you will model a mobile robot with four steerable wheels. Assume that each wheel has two actuators, one to steer it and one to drive it. The wheel locations relative to the chassis frame $\{\text{b}\}$ mimic the case of the four-wheeled robot in Figure 13.5: they are located at the four points $(\pm\ell, \pm w)$ in $\{\text{b}\}$. The steering angle θ_i of wheel i is zero when it rolls in the $+\hat{\text{x}}_\text{b}$-direction for a positive driving speed $u_i > 0$, and a positive rotation of the steering angle is defined as counterclockwise on the page. The linear speed at wheel i is ru_i, where r is the radius of the wheel.

(a) Given a desired chassis twist $\mathcal{V}_b$, derive equations for the four wheel steering angles θ_i and the four wheel driving speeds u_i. (Note that the pair (θ_i, u_i) yields the same linear motion at wheel i as $(-\theta_i, -u_i)$.)

(b) The "controls" for wheel i are the steering angle θ_i and the driving speed u_i. In practice, however, there are bounds on how quickly the wheel steering angle can be changed. Comment on the implications for the modeling, path planning, and control of a steerable-wheel mobile robot.