Chapter 2

Movable Machines

DRAFT OF CHAPTER 2 (8 Jan 2013)
(likely to be updated soon)

Mobile Robotics: An Information Space Approach

Steven M. LaValle, University of Illinois

Mobile robots combine three kinds of systems:

1. **Actuation:** Powered mechanical systems that cause motion. Examples are motors, thrusters, and propellers.
2. **Sensing:** Devices that output signals in response to external stimuli. Examples are cameras, laser scanners, and microphones.
3. **Computation:** One or more digital processors (CPUs) with memory. Examples are laptops, embedded processors, and microcontroller boards.

Actuation is the topic of this chapter: If a signal is sent to the actuation system, what happens? The robot should move in some desired way. Ideally, it should move predictably. In reality, it may move mostly as desired but with some unpredictable errors. In either case, we want to characterize what will happen so that we can add in the other two systems. Sensors will provide information that is used to decide whether to change the signals to the actuation system. Computation may be needed to process the sensor readings and all prior information, and make a decision that helps solve the task. Sensors and computation are the subjects of later chapters.

2.1 Mechanical Hardware

Robot locomotion is a general subject that addresses ways in which a mobile robot could move. Locomotion is an old branch robotics, yet remains one of the most

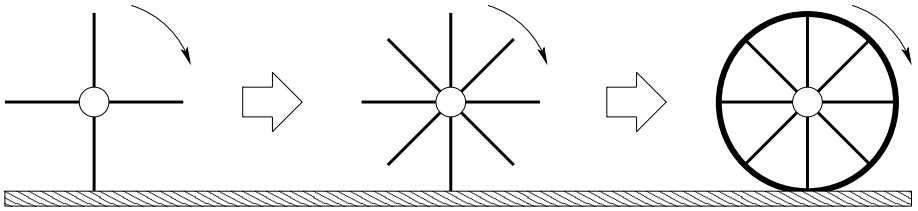


Figure 2.1: A few legs that clumsily "roll" along can be considered as a rimless wheel. Increasing the number of legs gradually produces a wheel.

active areas of research. The particular choice of locomotion method depends on many factors, such as:

1. **Medium:** What kind of environment will the robot traverse? Examples are rugged terrain, factory floors, air, space, on top of water, and under water.
2. **Stability:** Will the robot platform shake, vibrate, or tumble while moving? The robot might need to carry a camera that streams video without making viewers sick. Alternatively, perhaps the robot carries delicate materials.
3. **Payload:** How much weight will the robot be required to carry?
4. **Energy consumption:** Mobile robots are notoriously limited by their battery capacity. Low-energy methods of locomotion contribute greatly to their longevity.
5. **Durability:** Can the robot survive extended use in harsh conditions such as cold weather, snow, rain? Can the robot overcome collisions with obstacles and other robots?
6. **Cost:** The lower the cost, the greater the likelihood that the robot will be mass produced, which leads to greater impact on society.

Thousands of robot designs exist, which each address these factors in different ways. Chapter 1 showed robots that perform *legged locomotion* by walking through indoor and outdoor environments. Refer to Figures 1.5(e) and 1.6. An important design consideration is the *gait*, which specifies the motions of legs and the order in which feet are placed on the ground and removed from the ground. Ensuring stability while one or more feet are not touching the ground is a challenging problem. Other methods of locomotion that involve gaits include slithering snakes, the swimming robot of Figure 1.9(a), and the vibrating bug in Figure 1.13. Locomotion may also be achieved by *thrusting*, as in the case of the autonomous motor boat in Figure 1.9(b), the UAV in Figure 1.9(c), and the quad-rotor helicopter in Figure 1.10.

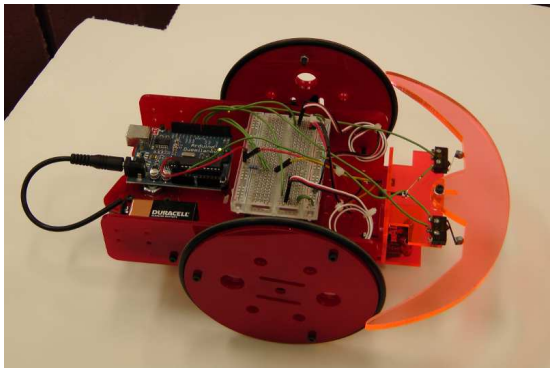


Figure 2.2: A low-cost robot built at the University of Illinois by extending the SERB open-source mobile design. The wheels and platform are cut from acrylic sheets. Each wheel is connected to a servo, and an Arduino microcontroller provides commands.

Finally, the oldest and most common form of locomotion was offered by the invention of the wheel. By comparing to the *rimless wheel* shown in Figure 2.1, *rolling locomotion* is the beautiful limiting case of having an infinite number of legs that continuously move in unison. This case will be the main focus for the remainder of this chapter; however, keep in mind that mathematical models of other locomotion methods have similar forms. Most of the robots from Section 1.1 move by rolling.

For rolling locomotion, each wheel is usually connected to an axle that is turned by a motor. A simple, low-cost design is shown in Figure 2.2. At the high end, virtually any car, truck, or construction vehicle that has been designed for human use has been retrofitted for automated driving. An example is the Google car from Figure 1.5(c). Large robotic vehicles at the high end use sophisticated electrical and mechanical systems that incorporate sensor feedback and gearing to deliver predictable, reliable performance. Most of these build on extensive automotive technology that has been developed over the past century. In addition, many unusual rolling locomotion systems have been designed specifically for mobile robots. For example, the *Mecanum (or Swedish) wheel* (Figure 2.3) can roll both forward as an ordinary wheel and perfectly sideways, which is quite unusual. This greatly increases the robot's mobility, an issue which becomes important in this chapter.

At the "unsophisticated" extreme, low-end robots can be constructed from simple components with a minimal amount of wiring and assembly skills. It is worthwhile to read popular, hobbyist robotics books (and build some robots!) to become familiar with widely available components []. Each wheel axle is normally attached to the armature of a battery-powered, DC (direct current) motor. *Stepper motors* are a common choice because the 360-degree rotation cycle is divided

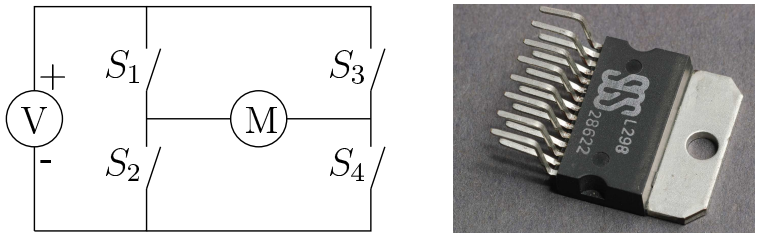
Figure 2.3: In 1985, the CMU URANUS mobile robot achieved omnidirectional rolling by driving four Mecanum wheels, which were invented by Bengt Ilon.

(a) Lego NXT servo (b) Hobby servos

Figure 2.4: Cheap, common servo motors: (a) A servo included in the Lego Mindstorm NXT robot kit. (b) A couple of servos that often appear on hobby vehicles, such as remote-controlled cars. (Source: Wikipedia)

into a predetermined number of steps. This allows more accurate prediction of the amount of wheel rotation. The system should be well calibrated so that the number of steps can be reliably commanded and predicted. A precise clock signal can be used to provide voltage pulses to the motor for prescribed time periods. Ultimately, performance is greatly improved by using a *servo motor* (or *servo*), which incorporates sensor feedback to ensure that the commanded amount of rotation has been achieved. Examples of commonly used servos are shown in Figure 2.4. Servos can alter the steering direction of a robot in addition to driving its wheels.

The remaining sections of this chapter describe the effects of commands that are given to the mobile robot. A command is a digital signal that is converted into wheel actuation using a circuit such as the H-bridge shown in Figure 2.5(a). The battery is connected to the motor, but there are four controllable switches. A cheap solid-state device, shown in Figure 2.5(b), is used to convert digital signals into opened or closed switch configurations. Table 2.1 describes the response of



(a) (b)

Figure 2.5: (a) An H-bridge circuit, which is a standard way to control a robot motor. The power source and motor are represented inside of circles labeled "V" and "M", respectively. There are four switches, S_1 to S_4, that can be independently *open* or *closed*. (b) A solid-state MOSFET device that implements the H-bridge.

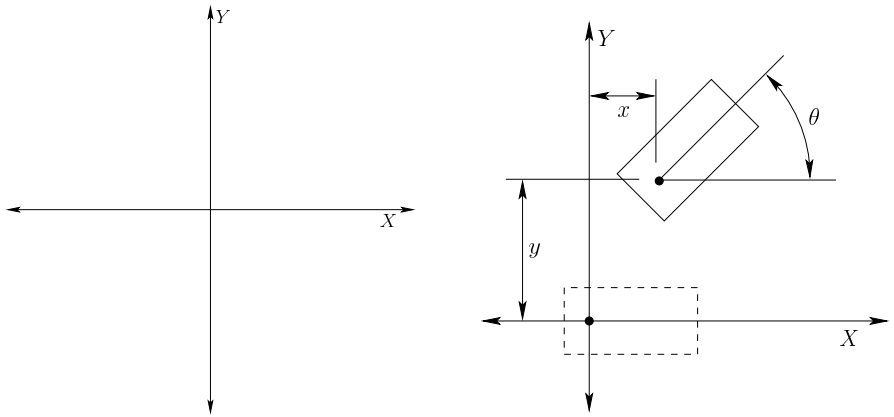
S_1	S_2	S_3	S_4	Result
1	0	0	1	Actuate wheel in forward direction
0	1	1	0	Actuate wheel in reverse
0	0	0	0	Allow the wheel to coast
0	1	0	1	Apply braking to stop the wheel quickly
1	1	0	0	Short circuit (useless)

Table 2.1: The effects of five switch configurations from the H-bridge circuit from Figure 2.5(a). A "0" indicates that the switch is *open*, and "1" indicates *closed*. The remaining nine possible configurations each produce one of these five results.

the motor for several configurations of the switches.

2.2 Kinematics of Perfect Rolling Robots

The section describes the *kinematics* of rolling robots. The term kinematics means the geometry of motion without describing the forces that cause it. Every robot in this section is modeled as a rigid platform that has attached wheels. Wheels are able to rotate, but some are driven by a motor and others just passively roll. In some cases, the direction that the wheel is pointing may also rotate with respect to the platform. If the direction is driven by a motor, then the robot can be steered. We want a precise account of where the robot will go when motors are turned on and the wheels are rolling. The robots are considered "perfect" in this section in the sense that they behave in a predictable way that can be described with simple equations. Section 2.4 covers the case in which they become less predictable due to bumpy surfaces, slipping wheels, dying batteries, and so on.



(a) (b)

Figure 2.6: (a) The mobile robot moves in an infinite plane, with positions specified by x and y coordinates. (b) When the robot is placed at configuration (x, y, θ), it is moved from its home configuration (depicted by dashed lines) into the shown position and orientation.

2.2.1 Single-Wheel Drive

To begin talking about the motion of a mobile robot, we will use the coordinate system shown in Figure 2.6(a). The robot moves along a "floor" that is the XY-plane and extends infinitely in all directions. The position and orientation of the robot platform will be called the *configuration*. To uniquely describe every possible configuration, we will use three parameters: x, y, and θ, which are written together as (x, y, θ). See Figure 2.6(b). The "home" configuration of the robot is $(0, 0, 0)$. The *special point* is a location on the robot that has coordinates $(0, 0)$ when the robot is at its home configuration. Any other configuration (x, y, θ) means that the special point on the robot is at position (x, y), and the robot is rotated by θ with respect to its home configuration. The x and y parameters may take any real value; however, to eliminate redundancy, θ is limited from 0 to 2π (including 0, but not 2π).

Any point (x_p, y_p) on the robot when it is in its home configuration will end up at

$$(x + x_p \cos\theta - y_p \sin\theta, \quad y + x_p \sin\theta + y_p \cos\theta) \tag{2.1}$$

if the robot is at configuration (x, y, θ). This is derived by first rotating the robot by θ and then translating (shifting) it by x in the X-direction and y in the Y direction. Using a 2×2 rotation matrix,

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \tag{2.2}$$

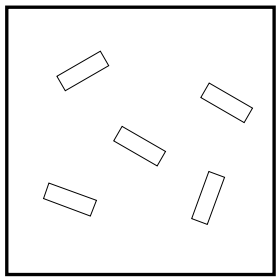


Figure 2.7: If wheels oriented in the directions shown, then is it possible for this platform to roll without any wheels skidding?

the rotation part can be written as

$$\begin{pmatrix} x_p \cos\theta - y_p \sin\theta \\ x_p \sin\theta + y_p \cos\theta \end{pmatrix} = R(\theta) \begin{pmatrix} x_p \\ y_p \end{pmatrix}. \tag{2.3}$$

To perform the translation part, x is added to the first coordinate, and y to the second. This yields (2.1).

Now consider attaching small wheels to the platform. Each wheel has a direction with respect to the X axis, when the robot is at configuration $(0, 0, 0)$. See Figure 2.7. Here is a fundamental kinematics question:

> If some wheels are placed at specific positions and directions on the platform, under what conditions is the platform capable of rolling?

If the wheels are not aligned in a proper way, then some wheels may roll while others are skidding. How do we ensure that *all* wheels will roll?

One of the easiest ways to ensure rolling is to point all wheels in the same direction, as shown in Figure 2.8(a). Many other wheel placements will work, however, by considering an important concept from kinematics called the *instantaneous center of rotation* (ICR). Draw a line in the XY plane that is centered on the wheel and is perpendicular to its direction. In other words, it is the axis about which the wheel rotates, but projected down into the XY plane. If all of the lines intersect at one point, then their common point of intersection is called the ICR.

Our fundamental kinematics has a simple answer. The platform will roll if and only if all lines are parallel or there is an ICR. Figure 2.8 shows examples. If the lines are parallel, then the robot will rotate along a straight line. If there is an ICR, then the robot will rotate along a circular arc. The ICR is the center of the circle that contains the arc. In the case of parallel lines, imagine that the ICR is infinitely far away, in a direction perpendicular to the lines, and the robot moves along a circle of "infinite radius". All of our rolling robot models will be constructed from the ICR principle.

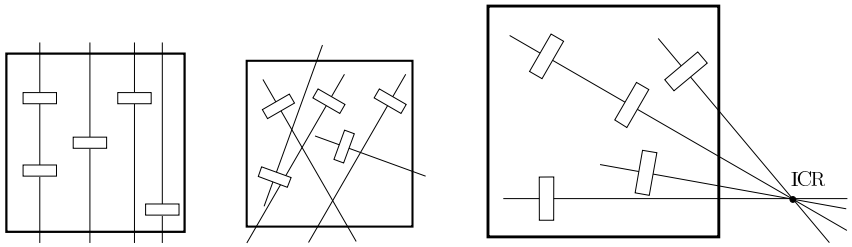


Figure 2.8: A top-down view of the wheels under a platform and their fixed directions: (a) If all wheel lines are parallel, then straight rolling occurs. (b) In this case, no instantaneous center of rotation (ICR) exists and the robot cannot roll. (c) The robot rolls with every wheel traversing a circle that is centered at the ICR.

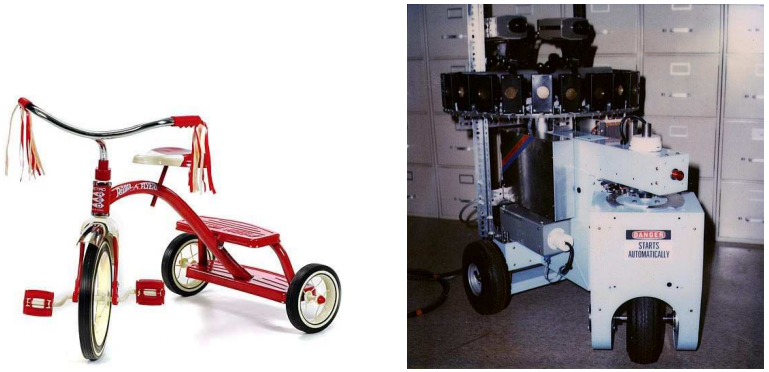


Figure 2.9: (a) A classic tricycle. (b) An old robot version: The Neptune robot at Carnegie Mellon University in 1984. Motors handle the "pedaling" and "steering".

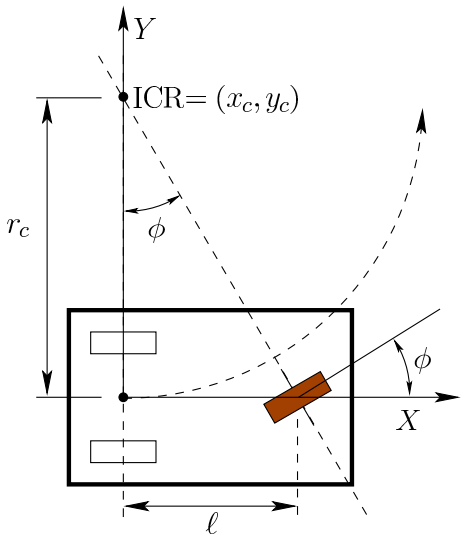


Figure 2.10: The important parameters of a tricycle.

For a simple illustration, consider pedaling a tricycle (Figure 2.9(a)). The front wheel is *powered* by pedaling and is *steered* by turning the handlebars. Figure 2.9(b) shows an old robotic equivalent of this. Figure 2.10 depicts the arrangement of wheels. The front wheel is shaded to indicate that it is powered. The white wheels rotate passively and are always directed forward. If the front wheel is pointing straight ahead, then clearly the robot moves straight and forward. If front wheel is turned left, as shown in Figure 2.10, then it traverses a circle, centered at the ICR.

We will now try to determine the configuration of the robot after it is driven. Suppose the robot starts at configuration $(0, 0, 0)$ and the front wheel is steered to the left by angle $\phi > 0$. See Figure 2.10. Let (x_c, y_c) denote the coordinates of the ICR. In this case, $x_c = 0$ and $y_c = \ell / \tan \phi$. To derive the position of y_c, note that a right triangle is formed by the line through the special point and the front wheel center, and the two lines perpendicular to the wheels. The base of the triangle is ℓ and the height is y_c. Its upper interior angle is ϕ; hence, from trigonometry $\tan \phi = \ell / y_c$. The triangle height is also the radius r_c of the circle traced out by the robot special point as it rolls. The circle is centered at the ICR: (x_c, y_c). Note that as ϕ decreases to zero, $r_c = y_c$ increases to infinity. In the limiting case when $\phi = 0$, the robot rolls straight, and the ICR does not exist (or is "at northern infinity").

If the front wheel rolls distance d_w along the circular arc, then what is the position and orientation of the robot? Another way to express this is to say that the wheel has rolled ψ radians and has radius r. In this case, $d_w = \psi r$, and we are back

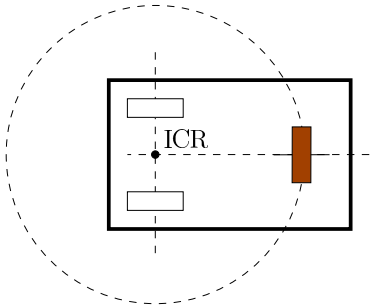


Figure 2.11: If the front wheel is turned to $\phi = \pi/2$, then the ICR coincides with the robot's special point and the robot rotates in place.

to the original question. The center of the front wheel is distance $r_w = \sqrt{\ell^2 + r_c^2}$ from the ICR. Hence, it moves along a circle of radius r_w, centered at (x_c, y_c). The back wheels, however, move along concentric circles of different radii because they are not distance r_w from the ICR. To determine the robot configuration, we need to determine where the special point moves to and how much the robot rotates. Let d_s denote the distance traveled along a circular arc by the special point. The special point travels less than the front wheel because $r_c < r_w$. The moving robot is a rigid body that rotates counterclockwise about the ICR. The distance therefore scales according to the difference in radii: $d_s = d_w r_c / r_w$. If the special point moves distance d_s along the circle of radius r_c, centered at (x_c, y_c), then it rotates by angle $\Delta\theta = d_s / r_c$ (this is just what fraction of the circumference was traversed, scaled by 2π). The position of the special point will be

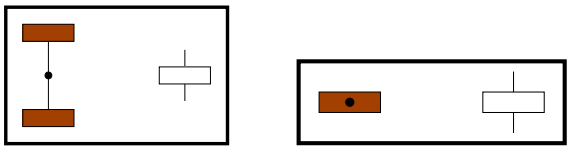
$$(x, y) = (r_c \sin \Delta\theta, y_c - \ r_c \cos \Delta\theta) \tag{2.4}$$

The resulting configuration is $(x, y, \Delta\theta)$.

If ϕ is negative, then the same method works, however, the robot will rotate clockwise, turning right the whole time. In this case, r_c is negative and the circle radius is actually $-r_c$. Any angle ϕ can be handled except $\pi/2$ or $-\pi/2$; see Figure 2.11. In this case, the ICR coincides with the special point. The front wheel traverses the circle of radius ℓ while the special point remains stationary and the back wheels rotate around it. The rotation is counterclockwise and $r_c = \ell$ in the case of $\phi = \pi/2$. It is clockwise and $r_c = -\ell$ in the case of $\phi = -\pi/2$. Note that if $|\phi| > \pi/2$, it is equivalent to rolling backwards, which is not a problem. For example, $\phi = 2\pi/3$ is equivalent to $\phi = -\pi/3$, but rotating the wheel in the opposite direction.

Now consider the more complicated case in which the robot does not start from its home configuration:

1. The robot starts in some generic configuration (x, y, θ).



(a) (b)

Figure 2.12: These two robot designs behave kinematically the same was the tricycle: (a) The rear wheels can instead be motorized; however, they must be coupled to roll together and have the same radius. (b) The pair of rear wheels can be replaced by a single wheel to obtain a bicycle; however, side-to-side balancing becomes an issue.

2. The front wheel is pointing ϕ degrees from the forward direction.

3. The front wheel drives distance d_w along the arc.

The task is to determine the resulting configuration, (x', y', θ'). The first problem is that the ICR must be transformed. The ICR at $(x_c, y_c) = (0, r_c)$ is rotated by θ and then translated by x and y. Again, $r_c = \ell / \tan \phi$ and may be positive, negative, or zero. The ICR when the robot is at configuration (x, y, θ) is then:

$$(x_c, y_c) = (x - r_c \sin \theta,\ y + r_c \cos \theta). \tag{2.5}$$

Note that for $(x, y, \theta) = (0, 0, 0)$, we obtain (2.4). The extension of driving along the circle from (2.4), is transformed to configuration (x, y, θ) to obtain

$$\begin{aligned} x' &= x_c + r_c \sin(\theta + \Delta\theta) \\ y' &= y_c - r_c \cos(\theta + \Delta\theta) \\ \theta' &= \theta + \Delta\theta. \end{aligned} \tag{2.6}$$

The model presented so far works for other wheel configurations. The rear wheels could be powered instead of the front wheel, as shown in Figure 2.12(a). In this case, careful motor coupling is needed to ensure that the wheels roll at the proper rate. If the robot is turning, then the inner wheel must rotate more slowly than the outer wheel. This problem is solved in automobiles by the *differential* mechanism. Figure 2.12 shows what happens if the two rear rears wheels are reduced to one: A bicycle is obtained. It rolls the same way as the tricycle, however, it must be capable of balancing. For a human-operated bicycle this is greatly helped by conservation of angular momentum.

2.2.2 Differential Drive

The most popular kinematic model for mobile robots is the *differential drive*, which avoids having to turn the wheels. Figure 2.13 depicts the wheel arrangement.

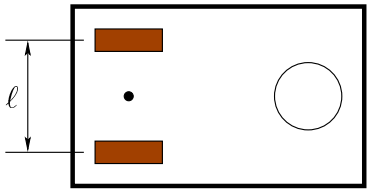
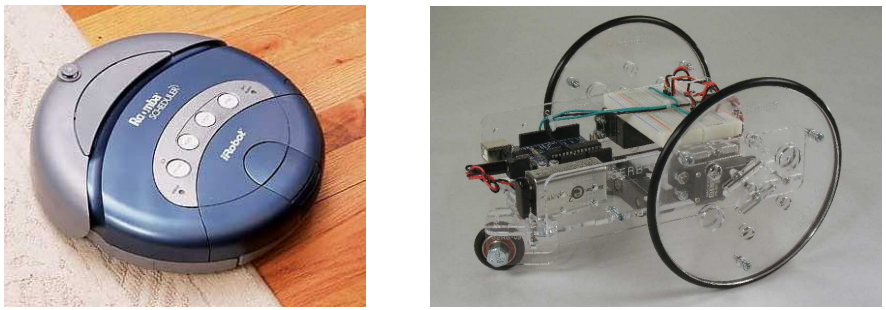


Figure 2.13: The differential-drive robot has independent motors that power each of the two rear wheels. A caster wheel is placed on the front for balance.



(a) (b)

Figure 2.14: Differential-drive robots: (a) The Roomba autonomous vacuum cleaner. (b) The Arduino Controlled Servo Robot (SERB) robot, which is a low-cost, open-source platform.

Figure 2.15: The Segway uses a differential drive to send standing users on their way. Rather than using an extra wheel for balance, a control system provides vertical stability.

Figures 2.14 and 2.15 show some real examples. For a differential drive, two non-steerable wheels are aligned so that a single perpendicular line crosses through both wheel centers. Both wheels have the same radius. Each wheel can be rotated independently. If they are rotated at different speeds, then the robot turns. There is usually a third wheel that simply keeps the robot from tipping over. A caster wheel is often used, as on the bottom of an office chair. The wheel orients itself to roll in any direction that it is forced.

Suppose that the two wheel motors are switched on so that each runs at a constant speed. Let d_l and d_r denote the distance rolled by the left and right wheels, respectively. The following simple motions arise:

1. If $d_l = d_r > 0$, then the robot rolls straight and forward.
2. If $d_l = d_r < 0$, then the robot rolls straight and backwards.
3. If $d_r > 0$ and $d_l = -d_r$, then the robot rotates in place counterclockwise.
4. If $d_r < 0$ and $d_l = -d_r$, then the robot rotates in place clockwise.

For the last two cases, how far does the robot rotate? Also, what happens in all other cases, meaning that $|d_l| \neq |d_r|$? These can be answered by determining

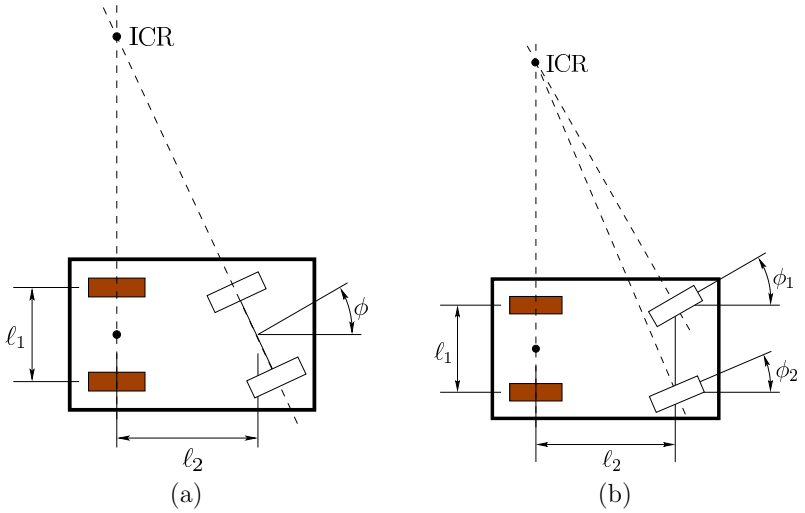


Figure 2.16: Car-like robots: (a) A car with a steered front axle. (b) A car with Ackerman steering, which is allows the front wheels to turn in place while maintaining the existence of the ICR.

the ICR and then applying the techniques from Section 2.2.1. It is clear that the ICR will lie along the common perpendicular line through the rear wheels and the special point.

All four of the simple motions above fit the following formula:

$$r_c = \frac{\ell(d_l + d_r)}{2(d_r - d_l)}. \tag{2.7}$$

In the case of $d_r = d_l$, then $r_c = \infty$, which implies that the robot moves straight (along a "circle with infinite radius"). The formula in (2.7) correctly specifies r_c for all other values of d_l and d_r, assuming that each motor rotates at a constant speed.

2.2.3 Car-Like Robots

Another common variant of the tricycle is the *car* or *car-like robot*. In this case there are two steered wheels in the front and two fixed wheels in the back. See Figure 2.16. The front wheels could be attached to a single steerable axle or they could be rotate in place, which is like virtually all automobiles. If they rotate in place, then the steering must be coupled in a special way so that their perpendicular lines both intersect the ICR. This is referred to as *Ackerman steering*. If both front wheels are instead turned at the same angle, then skidding would occur. As you may already know from automobiles, the power to the wheels could

be in the front, back, or both. This yields front-wheel drive, rear-wheel drive, and four-wheel drive, respectively.

The car moves in the same way as the tricycle. It is just a matter of figuring out the equivalent parameters. For the steered axle car in Figure 2.16(a), its motion is equivalent to having a single front wheel in the middle. The equations from Section 2.2.1 correctly describe the motion using the same ϕ and substituting $\ell_2 = \ell$. Now consider the Ackerman steered car in Figure 2.16(b). If there were a wheel in front center and it is turned angle ϕ, then the car would turn with radius $r_c = \ell_2 / \tan\phi$. To achieve the same radius using the Ackerman steering, the steering angles shown in Figure 2.16(b) must satisfy

$$r_c + \ell_1/2 = \ell_2 \tan\phi_1 \tag{2.8}$$

and

$$r_c - \ell_1/2 = \ell_2 \tan\phi_2. \tag{2.9}$$

Solving for ϕ_1 and ϕ_2 yields the steering angles that avoid skidding.

One important difference between the car-like robot and the other models has been overlooked so far. In a car, there is usually a limit on how far the wheels can be turned. Thus, if ϕ_{max} is the limit, with $\phi_{max} > 0$ and $\phi_{max} < \pi/2$, then the steering angle must be limited to $|\phi| \leq \phi_{max}$. This prevents the car from rotating in place, which is a capability of the tricycle and the differential drive. Instead, the car has a *minimum turning radius* $r_{min} = \ell_2 / \tan\phi_{max}$, which corresponds to the smallest circular arc that it can drive along.

2.3 Defining a Control System

A control system is a convenient mathematical way to express the effects of *commands* on the *states* of a electrical or mechanical system that evolves over time. In the context of mobile robotics, the command usually indicates which motors to activate, at what rate, for how long, and so on. This section merely describes the effect of the command, without regard to how it is chosen. That is the topic of coming chapters.

2.3.1 Discrete-Time Systems

Every control system has the following components:

1. An *command set* U, which represents every command that can be given to the system.[1]

2. A *state space* X, which characterizes every possible configuration or status of the system. Examples are coming soon!

[1]Common, alternative names for U are the *input set* or *action set*.

3. A *state transition equation*, which specifies how the state changes under the influence of a command.

In a *continuous-time* control system, the state transition equation is a differential equation. In a *discrete-time* control system, it instead provides the next state as a function of the current state and current command. Suppose $\mathbf{x}$ (which belongs to $\mathbf{X}$) is the current state and $\mathbf{u}$ (which belongs to $\mathbf{U}$) is a command that is applied to the system. The *discrete-time state transition equation* is

$$x' = f_d(x, u), \tag{2.10}$$

in which the function f provides the *next state* x' for every possible combination of $\mathbf{x}$ in $\mathbf{X}$ and $\mathbf{u}$ in $\mathbf{U}$. This idea appears in many other subjects, such as automata theory, finite state machines, and planning algorithms.

We now show take the kinematic models from Section 2.2 and make them into control systems. Recall the tricycle from Figure 2.10. What are the possible commands? The power to the wheel motor and the desired steering angle are natural candidates. To keep it simple at first, suppose that the motor is always rolling forward at some constant rate. The command is the steering angle $\mathbf{u} = \phi$. The command set is $\mathbf{U} = [-\pi, \pi)$.

Each state $\mathbf{x}$ is a configuration (x, y, θ) of the tricycle. The state space $\mathbf{X}$ represents all possible (x, y, θ). For the first two coordinates, any $(x, y) \in \mathbb{R}^2$ is possible. For θ, any value from 0 to 2π is possible; however, we must be careful to declare $\theta = 0$ and $\theta = 2\pi$ as the *same* orientation.

We now have enough notation to specify the state and command, but what does the "next state" mean? This means the state at some future point in time, usually when a special event occurs. In later chapters, this could correspond to the arrival of a special sensor reading. For example, it might indicate that the robot has hit an obstacle. In this chapter, we define it to mean the state after a predetermined amount $\Delta t > 0$ of time has passed.

The state transition equation $\mathbf{x}' = f_d(\mathbf{x}, \mathbf{u})$ means that if the tricycle starts in configuration $\mathbf{x} = (x, y, \theta)$ and command $\mathbf{u} = \phi$ is applied, then it will be in configuration $\mathbf{x}' = (x', y', \theta')$ after Δt seconds have elapsed. Assume that during the Δt time interval, the command cannot change. Figures 2.17 shows an example.

Figure 2.18 shows the effect of the commands from Figure 2.17. Let $\mathbf{x}_1$ denote the initial configuration. The first command $\mathbf{u}_1 = \pi/4$ is applied for time Δt, and the robot traverses a semicircle to the left, arriving in state $\mathbf{x}_2 = f_d(\mathbf{x}_1, \pi/4)$. Recall that Equation (2.6) tells how to get $\mathbf{x}_1$ from $\mathbf{x}_2$ if the robot rotates by θ_r, which in this case is $\theta_r = \pi$. At the next stage, $\mathbf{u}_2 = -\pi/4$ and the robot traverses a semicircle to its right, reaching $\mathbf{x}_3 = f_d(\mathbf{x}_2, -\pi/4)$. After that, $\mathbf{u}_3 = 0$, the the robot moves straight to reach $\mathbf{x}_4$. Finally, $\mathbf{u}_4$ is applied to yield the final state $\mathbf{x}_5$.

The discrete time model is conceptually simple. We apply a sequence of k commands:

$$(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k) \tag{2.11}$$

and the robot moves along k circular arcs or line segments. Some of its limitations are:

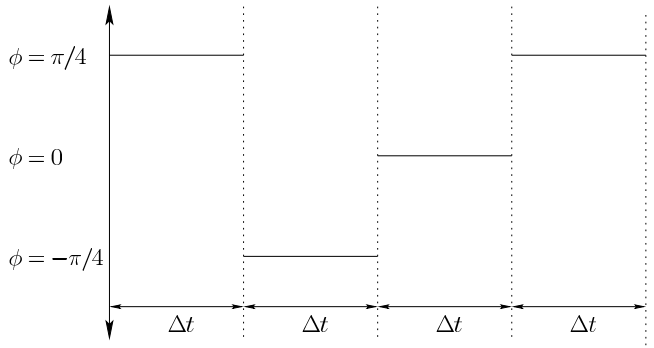


Figure 2.17: In our discrete-time model, the command is held constant over each Δt interval, and then it may switch to another command (or remain the same).

1. It may be too restrictive to hold the command constant over each Δt. The robot can move along a greater variety of curves if the command can vary continuously over time. Section 2.3 allows this by using a differential equation to express the state transition equation.

2. It is not physically feasible to instantaneously change the commands. For example, the wheel can not be changed from $\phi = 0$ to $\phi = \pi/4$ without going through all angles in between. Alternatively, the motor can not be instantaneously switched on to a desired speed with going through intermediate speeds. These problems are fixed by introducing more state variables, which is the subject of Section 2.3.3.

3. If we want to control the robot using the discrete-time model, then a clock or chronometer is needed to indicate precisely when Δt has elapsed so that the command is changed. We might instead want to change the command based on some other event, which is sensed. This becomes crucial in later chapters, to develop robots that respond to their sensor readings, rather than simply a Δt timer expiring.

2.3.2 Kinematics as a Differential Equation

Consider starting with a discrete-time model and making Δt very small. If some command $\mathbf{u}$ is applied, the robot will correspondingly move a small amount. Let $\Delta \mathbf{x} = x' - x$ denote the small change in the state. For our robots of Section 2.2, we write $\Delta \mathbf{x} = (\Delta x, \Delta y, \Delta \theta)$, meaning that x, y, and θ change by a small amount. Using the state transition equation (2.10),

$$\mathbf{x} + \Delta \mathbf{x} = f_d(\mathbf{x}, \mathbf{u}). \tag{2.12}$$

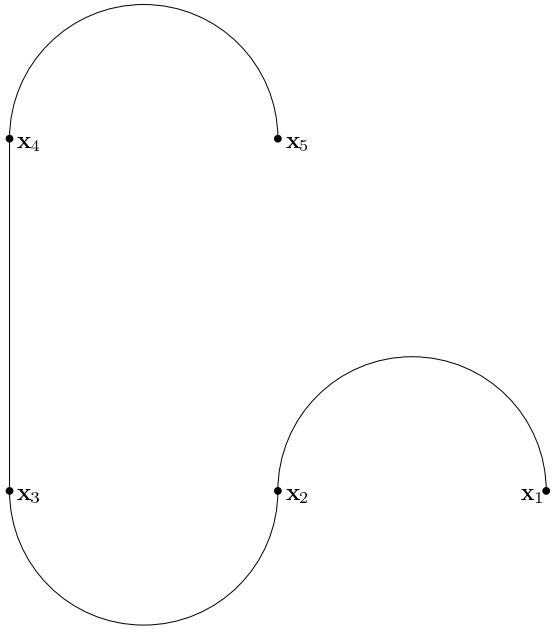


Figure 2.18: The special point on the robot traces out a path composed of circular arcs and line segments.

We want to develop a way to obtain the state in the limit as Δt approaches zero. Note that

$$\frac{\Delta \mathbf{x}}{\Delta t} \approx \frac{d\mathbf{x}}{dt}, \tag{2.13}$$

in which $d\mathbf{x}/dt$ is just the derivative with respect to time of every component of the state. For our robots,

$$\frac{d\mathbf{x}}{dt} = \begin{pmatrix} \frac{dx}{dt} & \frac{dy}{dt} & \frac{d\theta}{dt} \end{pmatrix}. \tag{2.14}$$

For convenience, we will place a dot "." over each variable to denote its time derivative. For example $\dot{\theta}$ means $d\theta/dt$.

For each value of $\mathbf{x}$ and $\mathbf{u}$, we want to know the corresponding $\dot{\mathbf{x}}$. This leads to a differential equation known as the *continuous-time state transition equation*:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}). \tag{2.15}$$

Once f is fully specified, the *state velocity* $\dot{\mathbf{x}}$ can always be determined. This can in turn be used to calculate future states via integration:

$$x(t) = x_0 + \int_0^t f(u(t), u(t)). \tag{2.16}$$

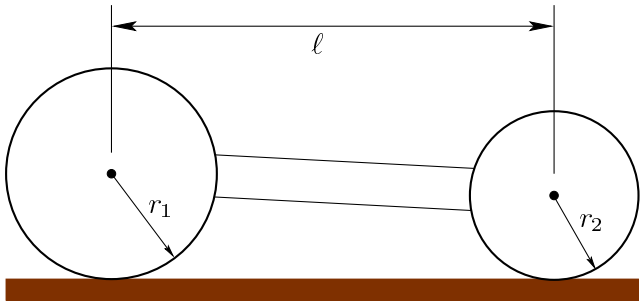


Figure 2.19: The side view of the bicycle. Forward rolling moves it to the left.

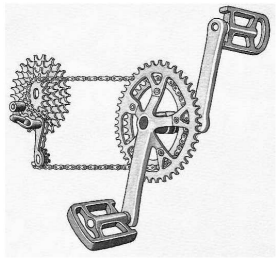


Figure 2.20: The wheel rotation rate is the gear ratio times the pedaling rotation rate. Similar gearing occurs in motors.

This expression uses the notation $x(t)$ and $u(t)$ to specify the state and command, respectively, at time t. This implicitly defines two time paramtrized functions.

To understand why (2.16) works, consider performing a rectangular approximation to the integral. Using the fact that $\Delta \mathbf{x} \approx \dot{\mathbf{x}} \Delta t$, we can approximate the integral in (2.16) as

$$\tilde{\mathbf{x}}(t) \approx x_0 + \sum_{i=0}^{k-1} f(\tilde{\mathbf{x}}(k\Delta t), \tilde{\mathbf{u}}(k\Delta t))\Delta t. \tag{2.17}$$

In each step, f is applied to determine the state velocity at time $k\Delta t$. In the limit as Δt tends to zero, (2.16) is obtained.

Now we specialize (2.15) for our rolling robots. First consider the bicycle with rear-wheel drive (Figure 2.12(b)). Figure 2.19 shows a side view of the wheels. A motor turns at some rate, such as 2 revolutions per second. For a bicycle with a chain (Figure 2.20) this is the rate at which you are pedaling (you are the "motor"). By using gears, the rate is multiplied by some constant factor to obtain the wheel rotation rate. For example, if the gear sprockets have a 3-to-1 ratio of teeth, then the wheel would turn at $2 \cdot 3 = 6$ revolutions per second. This means that the

rear wheel angle ψ_2 rotates through 12π radians in one second. The total distance that it rolls is $r_2\psi_2$. If the bicycle is not rolling straight, then it is the length of the traversed curve.

Let $\omega_2 = \dot{\psi}_2$, which is the *angular velocity* if the rear wheel. The linear speed of the robot special point, which is at the rear wheel center, is $s = r_2\omega_2$. If the robot orientation is θ, then the speed is divided into X and Y velocity components. For example, if $\theta = 0$, then $\dot{x} = s$. If $\theta = \pi/2$, then $\dot{y} = s$. If $\theta = \pi/4$, then $\dot{x} = \dot{y} = 1/\sqrt{2}$. The $\sqrt{2}$ appears because the speed s is the magnitude of the velocity $\sqrt{\dot{x}^2 + \dot{y}^2}$. In general, we obtain $\dot{x} = s\cos\theta$ and $\dot{y} = s\sin\theta$ from simple trigonometry. The state transition equation is therefore

$$\begin{aligned} \dot{x} &= s\cos\theta \\ \dot{y} &= s\sin\theta \\ \dot{\theta} &= \omega, \end{aligned} \tag{2.18}$$

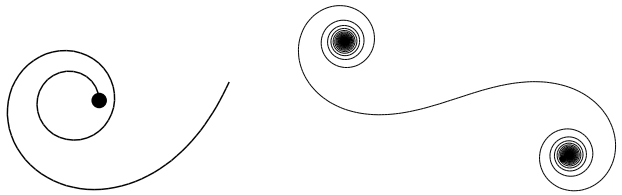
in which the third equation expresses the derivative of θ in terms of the robot's angular rate ω of rotation. Recall from Section 2.2 that if the steering angle is ϕ, then the robot moves along a circle of (signed) radius $r_c = \ell_2/\tan\phi$. To convert linear velocity into angular velocity, divide s by r_c to obtain:

$$\begin{aligned} \dot{x} &= s\cos\theta \\ \dot{y} &= s\sin\theta \\ \dot{\theta} &= \frac{s}{\ell}\tan\phi. \end{aligned} \tag{2.19}$$

This equation works for all cases except $|\phi| = \pi/2$. In this case, the rear-wheel drive bicycle cannot roll. If the front wheel were powered instead, then the rear wheel would be rotating about the Z axis (out of the plane!) with a single point touching the ground, rather than rolling.

Recall that the commands $\mathbf{u}$ and the command set $\mathbf{U}$ are critical parts of any control system. What is the command $\mathbf{u}$ for the control system in (2.19)? A reasonable choice is the vector $\mathbf{u} = (s, \phi)$. Since s can be derived from the rear wheel angular velocity ω_2, we could instead use $\mathbf{u} = (\omega_2, \phi)$ and express (2.19) in terms of ω_2. We could even define $\mathbf{u} = (\omega_m, \phi)$, in which ω_m is the motor's angular velocity, which is then converted into s by using the gear ratio and wheel radius. To specify $\mathbf{U}$, we need to give the limits on ϕ and s. For example, we can assert that $-\pi/2 < \phi < \pi/2$ and $-1 \leq s \leq 1$. This makes $\mathbf{U}$ into a rectangular subset of the plane: $\mathbf{U} = (-\pi/2, \pi/2) \times [-1, 1]$. Including $s = 0$ allows the robot to stop, and $s = -1$ allows it to move in reverse.

By using (2.19), the robot is no longer required to move in circular arcs and line segments. For example, if s is held constant at $s = 1$ and $\phi(t) = \pi e^{-t}/2$, then the robot will travel along a spiral. The particular curve is obtained by integrating (2.19), as shown in (2.16). This could be performed numerically using (2.17), which results in a piecewise-circular approximation to the spiral. Many methods exist that produce closer approximation. One of the most widely used is



(a) (b)

Figure 2.21: The continuous-time transition equation allows paths that are not circular arcs or line segments by varying the command variables. (a) Keeping constant speed but varying the steering angle gradually from $\pi/4$ to 0. (b) Gradually decreasing the steering angle from $\pi/2$ to $-\pi/2$.

4th-Order Runge-Kutta integration, which yields close approximations with little computational overhead. It is based on a higher-order Taylor-series approximation to the differential equation, rather than (2.17), which is first order and is called *Euler integration*. The result is

$$x(\Delta t) \approx x(0) + \frac{\Delta t}{6}(w_1 + 2w_2 + 2w_3 + w_4), \tag{2.20}$$

in which

$$\begin{aligned} w_1 &= f(x(0), u(0)) \\ w_2 &= f(x(0) + \tfrac{1}{2}\Delta t\ w_1,\ u(\tfrac{1}{2}\Delta t)) \\ w_3 &= f(x(0) + \tfrac{1}{2}\Delta t\ w_2,\ u(\tfrac{1}{2}\Delta t)) \\ w_4 &= f(x(0) + \Delta t\ w_3,\ u(\Delta t)). \end{aligned} \tag{2.21}$$

The transition equation in (2.18) holds the same for all of the other rolling robot models. The only problem is to determine precisely what s and ω depend on in terms of motor speeds, wheel radii, and distances between wheels. For the rear-wheel drive tricycle in Figure 2.12(a), the left and right wheels must be driven at different rates to ensure rolling. It is harmless to imagine that there is a third rear wheel placed midway between the two actual rear wheels. Therefore, we can reuse (2.19) by calculating the inner and outer rear wheel speeds that produce speed s at the center. Suppose $\phi > 0$. Using the same $r_c = \ell_2/\tan\phi$, we obtain $s(r_c - \ell_1/2)$ for the left wheel speed and $s(r_c + \ell_2/2)$ for the right wheel speed. This is derived by scaling the speed s to account for the different wheel radii with respect to the circle centered at the ICR. Note that if the tricycle turns sharply, then the wheels must be powered to rotate in opposite directions. This behavior is closely related to the differential drive (Figure 2.13) which will be revisited shortly. Note that the rear-powered tricycle can handle the case of $|\phi| = \pi/2$ by powering the rear wheels to rotate in opposite directions at the same speed.

For the bicycle, or the tricycle originally shown in Figure 2.10, the transition equation depends on the front wheel radius r_1 and angular velocity ω_1, rather than

r_2 and ω_2. Let s_w denote the speed of the front wheel, in the direction it is rolling. Note that $s_w = r_2\omega_2$. To determine s, the speed of the special point, it must be scaled in the same way as in Section 2.2.1. In that section, d_w was related to d_s as $d_s = d_w r_c/r_w$. Taking the time derivative of both sides yields $s = s_w r_c/r_w$. Substituting this expression into (2.19) yields the transition equation. With front-wheel drive, the tricycle can actually rotate in the case of $|\phi| = \pi/2$, and we obtain $\dot{x} = \dot{y} = 0$.

The car-like robot can also be adapted from (2.19). With the rear-wheel drive car, the equations are the same. The left and right wheel speeds need to be varied, as mentioned for the rear-powered tricycle. If the front wheels are attached to a steered axle (Figure 2.16(a)), then ϕ is used in the same way as for the bicycle. For Ackerman steering (Figure 2.16(b)), the particular ϕ_1 and ϕ_2 angles are given by (2.8) and (2.9).

The only remaining robot model is the differential drive (Figure 2.13). Suppose that both wheels had radius r. If the left and right wheels are powered at ω_l and ω_r, respectively, then

$$\begin{aligned} \dot{x} &= \frac{r}{2}(\omega_l + \omega_r)\cos\theta \\ \dot{y} &= \frac{r}{2}(\omega_l + \omega_r)\sin\theta \\ \dot{\theta} &= \frac{r}{\ell}(\omega_r - \omega_l). \end{aligned} \tag{2.22}$$

This is a special form of (2.18) in which $s = r(\omega_r + \omega_l)/2$ and $\omega = r(\omega_r - \omega_l)/\ell$. The translation speed depends on the average of the angular wheel velocities. To see this, consider the case in which one wheel is fixed and the other rotates. This initially causes the robot to translate at $1/2$ of the speed in comparison to both wheels rotating. The rotational speed $\dot{\theta}$ is proportional to the change in angular wheel speeds. The robot's rotation rate grows linearly with the wheel radius but reduces linearly with respect to the distance between the wheels.

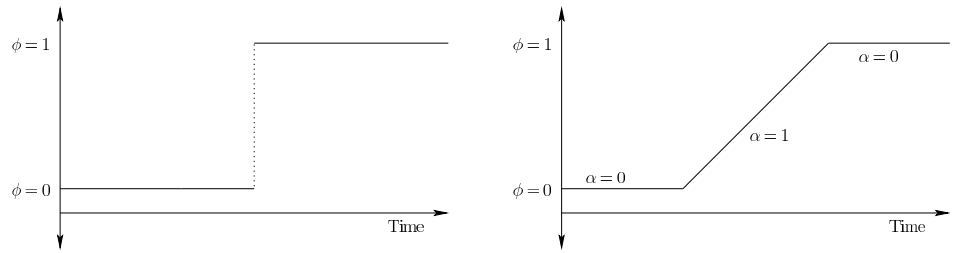
2.3.3 From Velocity to Acceleration

.

As mentioned at the end of Section 2.3.1, it is often unreasonable to assume that the command can be instantaneously switched without going through intermediate values. This problem is addressed by introducing more state variables. For example, the front wheel of the tricycle cannot be instantaneously turned to a new heading. To fix this problem, the state vector is extended to

$$\mathbf{x} = (x, y, \theta, \phi), \tag{2.23}$$

which means that ϕ is one of the state variables. This seems reasonable because the steering direction must be mechanically actuated. Instead of applying ϕ directly as a command, we introduce a new command variable $\alpha = \dot{\phi}$ that represents the rate



(a) (b)

Figure 2.22: (a) Using the previous transition equation (2.19) for the bicycle, the front wheel steering angle can be changed instantaneously, discontinuously switched from $\phi = 0$ to $\phi = 1$. (b) By defining ϕ as a new state variable in (2.25), we apply $\alpha = \dot{\phi} = 1$ as a command to bring ϕ from 0 to 1 continuously (visiting all intermediate angles).

of change of the steering angle. Suppose it is limited so that $|\alpha| \leq 1$. This means that the steering wheel can be turned no faster than 1 rad/sec^2. The command vector becomes:

$$\mathbf{u} = (\alpha, s) \tag{2.24}$$

and $\mathbf{U} = [-1, 1] \times [-1, 1]$. The state transition equation remains mostly the same as in (2.19), but gains a fourth component:

$$\begin{aligned} \dot{x} &= s \cos\theta & \dot{\theta} &= \frac{s}{\ell} \tan\phi \\ \dot{y} &= s \sin\theta & \dot{\phi} &= \alpha. \end{aligned} \tag{2.25}$$

For this model, the steering angle ϕ must rotate through all intermediate values, as shown in Figure 2.22.

The same criticism could be made about being able to start and stop rolling instantaneously. Therefore, we can introduce an *acceleration* command variable a with the relationship that $\dot{s} = a$. Suppose the acceleration is limited to $|a| = 1$. This leads to $\mathbf{u} = (\alpha, a)$, $\mathbf{U} = [-1, 1] \times [-1, 1]$, and

$$\begin{aligned} \dot{x} &= s \cos\theta & \dot{\phi} &= \alpha \\ \dot{y} &= s \sin\theta & \dot{s} &= a \\ \dot{\theta} &= \frac{s}{\ell} \tan\phi. \end{aligned} \tag{2.26}$$

What if the steering angle be instantaneously brought from $\alpha = 0$ to $\alpha = 1$? A new state variable can be introduced so that the steering angle is accelerated by β and $\dot{\alpha} = \beta$. This makes ϕ vary smoothly, rather than having the corners shown in Figure 2.22(b). The resulting control system has $\mathbf{u} = (\beta, a)$, $\mathbf{U} = [-1, 1] \times [-1, 1]$,

and

$$\begin{aligned} \dot{x} &= s \cos\theta & \dot{\phi} &= \alpha \\ \dot{y} &= s \sin\theta & \dot{s} &= a \\ \dot{\theta} &= \frac{s}{\ell} \tan\phi & \dot{\alpha} &= \beta. \end{aligned} \tag{2.27}$$

There is clearly no limit to this process of introducing intermediate state variables. The purpose is to provide smoother vehicle motions and satisfy the physical limitations of mechanical systems. The models in this section provide helpful illustrations, but they are not general enough express the full relationships between accelerations, velocities, and configuration variables in general. The equations of motion are usually derived using classical mechanics [], but can nevertheless be converted into the control-system form $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$. To develop a high-quality model of motion, there may be dozens of variables, which is common for industrial automotive simulations. For a car-like robot, for example, the configurations and velocities of each of the four wheels contribute to $\mathbf{x}$, in addition to the position and orientation of the car, its steering angle, and their velocities. If the car has a suspension system, then even more variables could be added. Furthermore, the state space, command set, and state transition approach has been developed and used extensively for many other classes of robotic vehicles, types of locomotion, and environments.

2.4 Kinematics of Misbehaving Rolling Robots

The robot models given so far assume that everything is perfect. All robot parameters, such as wheel radius, steering angle, and distances between wheels are assumed to be *exactly* right. The motors must turn at exactly their commanded rate, switching instantaneously on and off, if required. It is furthermore assumed that the robot moves along a perfectly flat and smooth surface. The wheels also must roll perfectly without any slipping, skidding, or tire compression. There cannot be any dirt or other small imperfections between the tire-floor contact; otherwise, it would violate the assumptions. The perfect models are nevertheless valuable because they provide nominal behavior for the robot, which is used for designing strategies and mathematically analyzing motion strategies.

The perfect models also provide a good starting point upon which to add disturbances that account for an unpredictable robot operating in the physical world. Although it is easy to criticize the perfect models, it remains a difficult challenge to define good models of the imperfections. One of the first steps is *measurement and calibration*. The parameters, such as wheel radius, should be measured as accurately as possible. If the wheel radius is measured at 5cm when it is actually very close to 4.7cm, then the robot will systematically roll more slowly than the model predicts. This error should not be accounted for as random noise because it is a one-time mistake that consistently yields the same behavior for the life of the robot. It is best to eliminate as much of it as possible.

Calibration is used to make sure that the commands $\mathbf{u}$ are accurate. Suppose $\mathbf{u} = (s, \phi) = (1, 0)$ in (2.19), which means that the robot commanded to move straight with unit speed. If experiments are performed and the robot is observed to turn slightly left each time, then the steering angle should be shifted. Perhaps $\phi' = \phi - \epsilon$ should be used, in which $\epsilon > 0$ accounts for the amount of bias to the left. In other words, when the command to go straight $\phi = 0$ is applied, the robot is observed to turn as if $\phi = \epsilon$ were given. A similar calibration procedure should be used for the motor to make sure its rotation rate is correctly commanded. This in combination with proper measurements should make the s command as accurate as possible.

If the perfect model is still not accurate enough for practical use, then some additional modeling is needed. Suppose it is possible to place the robot in typical environments and conduct motion experiments. Starting configurations are measured, commands are given, and the resulting final configurations are measured. Motion capture systems can be used in a laboratory setting to automatically measure robot configurations. After sufficiently many trials, an empirical model of the unpredictable aspects could be developed. At another extreme, physical models of wheel slippage, bumpy surfaces, nonlinear motor characteristics, and so on can be developed to make more accurate mathematical models. An attempt can be made to explain the observed laboratory behavior with better models.

If this fails to provide sufficient performance, then extra disturbance parameters may be added to the model to account for all other problems. This leads to two popular choices:

1. **Nondeterministic:** Upper bounds are placed on the amount of deviation that is allowed from the perfect model. This specifies a *set* of possible future outcomes. For example, if the robot is commanded to go straight, it will stay within a narrow cone that faces straight ahead.

2. **Probabilistic:** A probability density function is defined over the state space. This also provides a set of possible future outcomes, but they are weighted by *probability*. More likely outcomes are given more weight, which hopefully matches observations made in a laboratory setting.

The nondeterministic choice usually leads to *worst-case analysis* by developing motion strategies that ensure that the system functions correctly in all possible outcomes. It is often called *robust*. The probabilistic choice instead leads to *expected-case* or *average-case analysis*. A strategy is then designed to optimize performance. For example, to arrive at the goal as quickly as possible in the average case.

2.4.1 Nondeterministic Control Systems

For the nondeterministic choice, imagine that we are playing a game against nature. Whenever we send a command to the robot, "nature" issues its own command, which interferes with the outcome. There is no way to precisely know what

command nature will choose, but its set of possible commands is at least assumed to be known. Let each command of nature be called a *disturbance*, $\mathbf{w}$. Each disturbance $\mathbf{w}$ is selected from a known *disturbance set* $\mathbf{W}(\mathbf{u})$. It is written as $\mathbf{W}(\mathbf{u})$ and not $\mathbf{W}$ to allow the possible disturbances to depend on the robot's command $\mathbf{u}$. This gives nature an advantage in the game. We can even write $\mathbf{W}(\mathbf{x}, \mathbf{u})$, which allows the disturbance set to depend on the state $\mathbf{x}$ as well.

A *nondeterministic control system* is defined by extending the discrete-time state transition equation (2.10) to account for the disturbance:

$$\mathbf{x}' = f_d(\mathbf{x}, \mathbf{u}, \mathbf{w}). \tag{2.28}$$

For a useful example, consider extending (2.6) to obtain a front-wheel-drive tricycle that receives interference from nature. The commands are the motor velocity and steering angle ϕ. Let $\mathbf{w} = (w_1, w_2)$, in which w_2 interferes with the speed and w_2 interferes with the steering. After nature chooses its command $\mathbf{w}$, the resulting radius r_c and change in orientation $\Delta\theta$ become

$$\tilde{r}_c = \ell / \tan(\phi + w_2) \tag{2.29}$$

and

$$\tilde{\Delta}\theta = (d_s + w_1)/\tilde{r}_c. \tag{2.30}$$

The symbol $\tilde{}$ is placed above variables to denote their disturbed values. We can write $\tilde{s} = s + w_1$, $\tilde{\phi} = \phi + w_2$, $\tilde{r} = \tilde{s}/\tilde{\omega}$ and $\Delta\tilde{\theta} = \tilde{s}\Delta\omega$. The disturbances result in a version of (2.6) in which $\tilde{r}$ and $\Delta\tilde{\theta}$ are substituted for r_c and $\Delta\theta$, respectively:

$$\begin{aligned} x' &= x_c + \tilde{r}_c \sin(\theta + \Delta\tilde{\theta}) \\ y' &= y_c - \tilde{r}_c \cos(\theta + \Delta\tilde{\theta}) \\ \theta' &= \theta + \Delta\tilde{\theta}. \end{aligned} \tag{2.31}$$

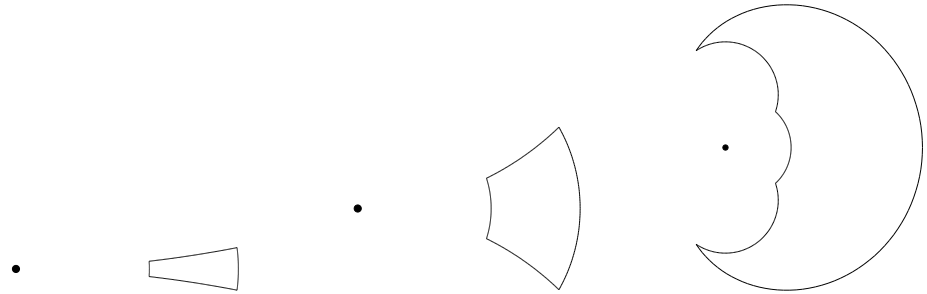
The disturbances w_1 and w_2 can even be inserted directly into the differential equation (2.19) to obtain

$$\begin{aligned} \dot{x} &= (s + w_1) \cos\theta \\ \dot{y} &= (s + w_1) \sin\theta \\ \dot{\theta} &= \frac{(s + w_1)}{\ell} \tan(\phi + w_2). \end{aligned} \tag{2.32}$$

This can be written equivalently as

$$\begin{aligned} \dot{x} &= \tilde{s} \cos\theta \\ \dot{y} &= \tilde{s} \sin\theta \\ \dot{\theta} &= \frac{\tilde{s}}{\ell} \tan(\tilde{\phi}). \end{aligned} \tag{2.33}$$

Similar disturbances can be inserted into other models, such as the differential drive or car-like robot.



(a) (b) (c)

Figure 2.23: Three forward projections for the nondeterministic tricycle: (a) Inaccurate speed. (b) Inaccurate steering angle. (c) Both speed and steering angle are highly inaccurate.

Because it is impossible to know which disturbance $\mathbf{w}$ nature will choose, it is useful to express the set of possible next states that could result if command $\mathbf{u}$ is applied from state $\mathbf{x}$. The result is called a *forward projection*, which describes possible futures. If $\mathbf{u}$ is applied to $\mathbf{x}$, then using (2.28), we can write

$$\mathbf{X}'(\mathbf{x}, \mathbf{u}) = \{\mathbf{x}' \in \mathbf{X} \mid \exists \mathbf{w} \in \mathbf{W}(\mathbf{u}) \text{ such that } \mathbf{x}' = f_d(\mathbf{x}, \mathbf{u}, \mathbf{w})\}, \tag{2.34}$$

which is the subset of $\mathbf{X}$ that may possibly be reached after some time Δt. Figure 2.23 shows some forward projections for the tricycle model.

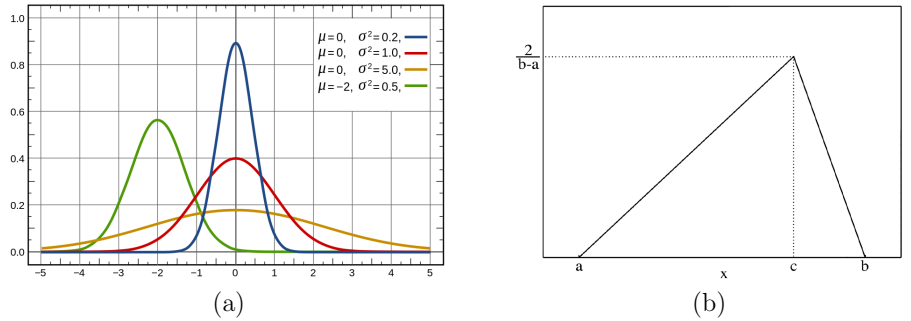
Starting from some $\mathbf{x}$ in $\mathbf{X}$, what possible future states could arise if $\mathbf{u}_1$ is applied, followed by $\mathbf{u}_2$, followed by $\mathbf{u}_3$? The forward projection (2.34) is iterated three times to yield:

$$\mathbf{X}'(\mathbf{X}'(\mathbf{X}'(\mathbf{x}, \mathbf{u}_1), \mathbf{u}_2), \mathbf{u}_3). \tag{2.35}$$

2.4.2 Probabilistic Control Systems

The probabilistic model is a direct extension of the nondeterministic model. The only difference is that weights are assigned to each disturbance $\mathbf{w}$ in $\mathbf{W}(\mathbf{u})$ to reflect its likelihood of occurring. Ideally, this assignment should be based on statistical models that are obtained by driving the robot numerous times and observing its behavior. In other words, the probabilistic model is *learned* through systematic experimentation. In some settings this may be too costly or time consuming. In others, it may be impossible. Therefore, heuristic models are often introduced to plausibly account for the disturbances and to allow for efficient probabilistic computations.

For any variable w, let $p(w)$ denote a *probability density function* (or *pdf*) over x. Figure 2.24 shows two examples. The most commonly used pdf is that of the



(a) (b)

Figure 2.24: (a) The probability density function (pdf) for the normal (Gaussian) distribution. (b) The pdf for a triangular density function. (Figures courtesy of Wikipedia.)

normal (or *Gaussian*) distribution. The precise expression is

$$p(w) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{w-\mu}{\sigma}\right)^2}, \tag{2.36}$$

in which μ is the *mean* and σ^2 is the variance. The normal distribution produces the familiar "bell curve" and arises repeatedly in probability and statistics.

Sometimes it will be helpful to allow the disturbance to depend on the chosen command $\mathbf{u}$. Whereas $\mathbf{W}(\mathbf{u})$ was used to represent the set of possible disturbances for each possible $\mathbf{u}$ in $\mathbf{U}$, we write $p(\mathbf{w}|\mathbf{u})$ to represent the corresponding probabilistic version, which is a conditional probability density function. For example, if the standard deviation σ grows linearly with some command u, then (2.36) would be replaced by

$$p(w|u) = \frac{1}{u\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{w-\mu}{u\sigma}\right)^2}, \tag{2.37}$$

to obtain a normal pdf for each value of u.

It might even be the case that $\mathbf{w}$ depends on both $\mathbf{x}$ and $\mathbf{u}$, which leads to a more general $\mathbf{W}(\mathbf{x}, \mathbf{u})$. However, for the simple kinematic models, it is unlikely that the particular disturbance depends on the configuration $\mathbf{x} = (x, y, \theta)$. Such invariance with respect to position and orientation greatly simplifies the task of learning the density $p(\mathbf{w}|\mathbf{u})$. The probabilistic counterpart of $\mathbf{X}(\mathbf{x}, \mathbf{u})$ is the pdf $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$. This is a density over possible next states $\mathbf{x}'$, after the command $\mathbf{u}$ is applied from state $\mathbf{x}$.

The disturbance $\mathbf{w}$ is applied to the transition equation $\mathbf{x}' = f_d(\mathbf{x}, \mathbf{u}, \mathbf{w})$ in the same way as in Section 2.4.1, but now each $\mathbf{w}$ carries a weight, $p(\mathbf{w}|\mathbf{x}, \mathbf{u})$. Once $\mathbf{x}$ and $\mathbf{u}$ are given, if the mapping from $\mathbf{w}$ to $\mathbf{x}'$ using f_d is one to one, then $p(\mathbf{x}'|\mathbf{x}, \mathbf{u}) = p(\mathbf{w}|\mathbf{x}, \mathbf{u})$. To determine the pdf value $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$, just determine the disturbance $\mathbf{w}$ that causes $\mathbf{x}'$ and apply $p(\mathbf{w}|\mathbf{x}, \mathbf{u})$. If multiple $\mathbf{w}$ values could

cause the same $\mathbf{x}'$ (with $\mathbf{x}$ and $\mathbf{u}$ already fixed), then it becomes slightly more complicated. The pdf values are summed if there is a finite number of $\mathbf{w}$ values or integrated if there is an infinite number.

One particular concern is that pdfs often have unbounded domain, as in the case of a Gaussian over all of $\mathbb{R}$, but are applied to bounded variables such as θ. For example, an unlikely disturbance could be 100π. If applied directly to θ, the resulting angle is $\theta + 100\pi = \theta$. To avoid this wraparound effect, the Gaussian domain may be truncated at $\pm\pi$ with negligible numerical effect. Otherwise, a bounded distribution, such as the triangular distribution of Figure 2.24(b) may be more appropriate. A more rigorous way to handle the angular disturbance problem is to use the *von Mises distribution* []:

$$p(w) = \frac{1}{I_0(\kappa)2\pi} e^{\kappa \cos(w-\mu)}, \tag{2.38}$$

in which I_0 is the modified Bessel function of order 0. The parameters μ and $1/\kappa$ are analogous to analogous to μ and σ^2, respectively, in (2.36).

The density $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$ can be considered as a one-step probabilistic forward projection, which is the counterpart to the nondeterministic forward projection $X'(\mathbf{x}, \mathbf{u})$. It can be iterated for obtain a multistep forward projection, as in the nondeterminstic case. A two-step probabilistic forward projection is expressed as:

$$p(\mathbf{x}_3|\mathbf{x}_1, \mathbf{u}_1, \mathbf{u}_2) = \int_{\mathbf{x}_2} p(\mathbf{x}_3|\mathbf{x}_2, \mathbf{u}_2) p(\mathbf{x}_3|\mathbf{x}_1, \mathbf{u}_1). \tag{2.39}$$

Computing further forward projections requires nested summations, which marginalize all of the intermediate states. For example, the three-stage forward projection is

$$p(\mathbf{x}_4|\mathbf{x}_1, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3) = \int_{\mathbf{x}_2} \int_{\mathbf{x}_3} p(\mathbf{x}_4|\mathbf{x}_3, \mathbf{u}_3) p(\mathbf{x}_3|\mathbf{x}_2, \mathbf{u}_2) p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{u}_1). \tag{2.40}$$

This is the probabilistic counterpart to (2.35).

Ordinarily, it is impossible to compute the integrals in closed form. Therefore, numerical methods are used. The integrals could be evaluated, for example, over a high-resolution grid of states. Alternatively, Monte Carlo methods can be used to generate sample paths by simulating the behavior of nature. In each Δt, a sample disturbance $\mathbf{w}$ is drawn from the probability density. Many simple methods exist to draw such samples. One of the most popular is the Box-Muller method. Suppose w_i follows a Gaussian with mean μ and variance σ^2. Using a standard random function that appears in most programming languages, draw u_1 and u_2 at random over the interval $[0, 1]$. The Box-Muller method generates a Gaussian sample with mean μ and variance σ^2 as

$$w_i = \sqrt{-2 \ln u_1} \cos(2\pi u_2)\sigma + \mu. \tag{2.41}$$

To generate one sample application of (2.31), w_1 and w_2 are generated using (2.41). These are transformed into $\tilde{r}_c$ and $\tilde{\theta}$ using (2.29) and (2.30). The next configuration (x', y', θ') is then generated using (2.31). Figure 2.25 shows a collection of samples obtained after several iterations of sampling disturbances.

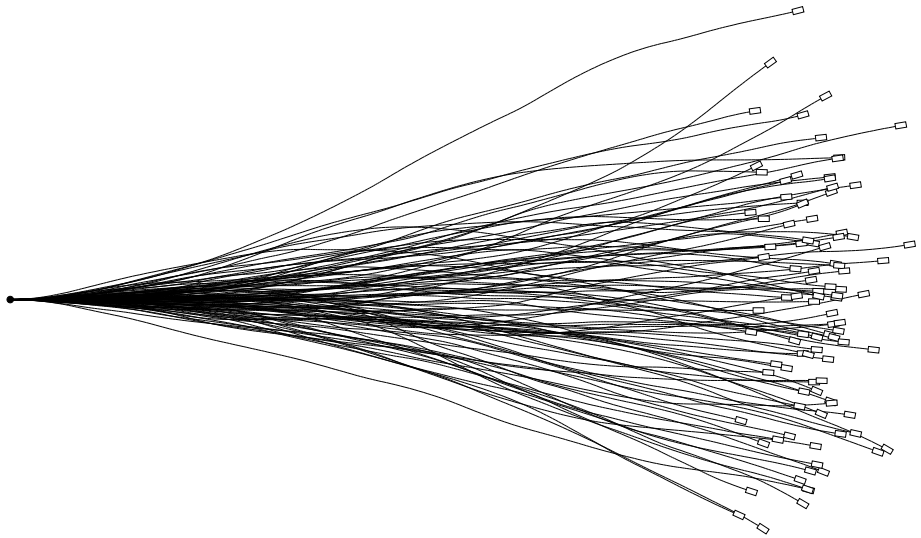


Figure 2.25: A sample set of 100 paths, obtained by iterating the tricycle model with Gaussian disturbance for 25 Δt steps.