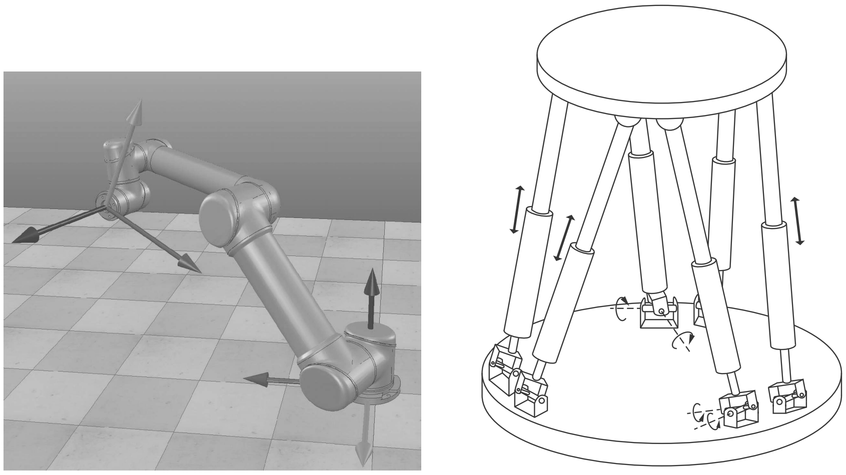
1 Preview

As an academic discipline, robotics is a relatively young field with highly ambitious goals, the ultimate one being the creation of machines that can behave and think like humans. This attempt to create intelligent machines naturally leads us first to examine ourselves – to ask, for example, why our bodies are designed the way they are, how our limbs are coordinated, and how we learn and perform complex tasks. The sense that the fundamental questions in robotics are ultimately questions about ourselves is part of what makes robotics such a fascinating and engaging endeavor.

Our focus in this book is on mechanics, planning, and control for **robot mechanisms**. Robot arms are one familiar example. So are wheeled vehicles, as are robot arms mounted on wheeled vehicles. Basically, a mechanism is constructed by connecting rigid bodies, called **links**, together by means of **joints**, so that relative motion between adjacent links becomes possible. **Actuation** of the joints, typically by electric motors, then causes the robot to move and exert forces in desired ways.

The links of a robot mechanism can be arranged in serial fashion, like the familiar open-chain arm shown in Figure 1.1(a). Robot mechanisms can also have links that form closed loops, such as the Stewart–Gough platform shown in Figure 1.1(b). In the case of an open chain, all the joints are actuated, while in the case of mechanisms with closed loops, only a subset of the joints may be actuated.

Let us examine more closely the current technology behind robot mechanisms. The links are moved by actuators, which typically are electrically driven (e.g., by DC or AC motors, stepper motors, or shape memory alloys) but can also be driven by pneumatic or hydraulic cylinders. In the case of rotating electric motors, these would ideally be lightweight, operate at relatively low rotational speeds (e.g., in the range of hundreds of RPM), and be able to generate large forces and torques. Since most currently available motors operate at low torques and at up to thousands of RPM, speed reduction and torque amplification are required. Examples of such transmissions or transformers include gears, cable drives, belts and pulleys, and chains and sprockets. These speed-reduction devices should have zero or low slippage and **backlash** (defined as the amount of rotation available at the output of the speed-reduction device without motion at



(a) An open-chain industrial manipulator, visualized in V-REP (Rohmer et al., 2013).

(b) Stewart–Gough platform. Closed loops are formed from the base platform, through the legs, through the top platform, and through the legs back to the base platform.

Figure 1.1 Open-chain and closed-chain robot mechanisms.

the input). Brakes may also be attached to stop the robot quickly or to maintain a stationary posture.

Robots are also equipped with sensors to measure the motion at the joints. For both revolute and prismatic joints, encoders, potentiometers, or resolvers measure the displacement and sometimes tachometers are used to measure velocity. Forces and torques at the joints or at the end-effector of the robot can be measured using various types of force–torque sensors. Additional sensors may be used to help localize objects or the robot itself, such as vision-only cameras, RGB-D cameras which measure the color (RGB) and depth (D) to each pixel, laser range finders, and various types of acoustic sensor.

The study of robotics often includes artificial intelligence and computer perception, but an essential feature of any robot is that it moves in the physical world. Therefore, this book, which is intended to support a first course in robotics for undergraduates and graduate students, focuses on mechanics, motion planning, and control of robot mechanisms.

In the rest of this chapter we provide a preview of the rest of the book.

Chapter 2: Configuration Space

As mentioned above, at its most basic level a robot consists of rigid bodies connected by joints, with the joints driven by actuators. In practice the links may not be completely rigid, and the joints may be affected by factors such as elasticity, backlash, friction, and hysteresis. In this book we ignore these effects for the most part and assume that all links are rigid.

With this assumption, Chapter 2 focuses on representing the **configuration** of a robot system, which is a specification of the position of every point of the robot. Since the robot consists of a collection of rigid bodies connected by joints, our study begins with understanding the configuration of a rigid body. We see that the configuration of a rigid body in the plane can be described using three variables (two for the position and one for the orientation) and the configuration of a rigid body in space can be described using six variables (three for the position and three for the orientation). The number of variables is the number of **degrees of freedom** (dof) of the rigid body. It is also the dimension of the **configuration space**, the space of all configurations of the body.

The dof of a robot, and hence the dimension of its configuration space, is the sum of the dof of its rigid bodies minus the number of constraints on the motion of those rigid bodies provided by the joints. For example, the two most popular joints, revolute (rotational) and prismatic (translational) joints, allow only one motion freedom between the two bodies they connect. Therefore a revolute or prismatic joint can be thought of as providing five constraints on the motion of one spatial rigid body relative to another. Knowing the dof of a rigid body and the number of constraints provided by joints, we can derive **Grübler's formula** for calculating the dof of general robot mechanisms. For **open-chain** robots such as the industrial manipulator of Figure 1.1(a), each joint is independently actuated and the dof is simply the sum of the freedoms provided by each joint. For **closed chains** like the Stewart–Gough platform in Figure 1.1(b), Grübler's formula is a convenient way to calculate the dof. Unlike open-chain robots, some joints of closed chains are not actuated.

Apart from calculating the dof, other configuration space concepts of interest include the **topology** (or "shape") of the configuration space and its **representation**. Two configuration spaces of the same dimension may have different shapes, just like a two-dimensional plane has a different shape from the two-dimensional surface of a sphere. These differences become important when determining how to represent the space. The surface of a unit sphere, for example, could be represented using a minimal number of coordinates, such as latitude and longitude, or it could be represented by three numbers (x, y, z) subject to the constraint $x^2 + y^2 + z^2 = 1$. The former is an **explicit parametrization** of the space and the latter is an **implicit parametrization** of the space. Each type of representation has its advantages, but in this book we will use implicit representations of configurations of rigid bodies.

A robot arm is typically equipped with a hand or gripper, more generally called an **end-effector**, which interacts with objects in the surrounding world. To accomplish a task such as picking up an object, we are concerned with the configuration of a reference frame rigidly attached to the end-effector, and not necessarily the configuration of the entire arm. We call the space of positions and orientations of the end-effector frame the **task space** and note that there is not a one-to-one mapping between the robot's configuration space and the task space. The **workspace** is defined to be the subset of the task space that the end-effector frame can reach.

Chapter 3: Rigid-Body Motions

This chapter addresses the problem of how to describe mathematically the motion of a rigid body moving in three-dimensional physical space. One convenient way is to attach a reference frame to the rigid body and to develop a way to quantitatively describe the frame's position and orientation as it moves. As a first step, we introduce a 3×3 matrix representation for describing a frame's orientation; such a matrix is referred to as a **rotation matrix**.

A rotation matrix is parametrized by three independent coordinates. The most natural and intuitive way to visualize a rotation matrix is in terms of its **exponential coordinate** representation. That is, given a rotation matrix R, there exists some unit vector $\hat{\omega} \in \mathbb{R}^3$ and angle $\theta \in [0, \pi]$ such that the rotation matrix can be obtained by rotating the identity frame (that is, the frame corresponding to the identity matrix) about $\hat{\omega}$ by θ. The exponential coordinates are defined as $\omega = \hat{\omega}\theta \in \mathbb{R}^3$, which is a three-parameter representation. There are several other well-known coordinate representations, e.g., Euler angles, Cayley–Rodrigues parameters, and unit quaternions, which are discussed in Appendix B.

Another reason for focusing on the exponential description of rotations is that they lead directly to the exponential description of rigid-body motions. The latter can be viewed as a modern geometric interpretation of classical screw theory. Keeping the classical terminology as much as possible, we cover in detail the linear algebraic constructs of screw theory, including the unified description of linear and angular velocities as six-dimensional **twists** (also known as **spatial velocities**), and an analogous description of three-dimensional forces and moments as six-dimensional **wrenches** (also known as **spatial forces**).

Chapter 4: Forward Kinematics

For an open chain, the position and orientation of the end-effector are uniquely determined from the joint positions. The **forward kinematics** problem is to find the position and orientation of the reference frame attached to the end-effector given the set of joint positions. In this chapter we present the **product of exponentials (PoE)** formula describing the forward kinematics of open chains. As the name implies, the PoE formula is directly derived from the exponential coordinate representation for rigid-body motions. Aside from providing an intuitive and easily visualizable interpretation of the exponential coordinates as the twists of the joint axes, the PoE formula offers other advantages, like eliminating the need for link frames (only the base frame and end-effector frame are required, and these can be chosen arbitrarily).

In Appendix C we also present the Denavit–Hartenberg (D–H) representation for forward kinematics. The D–H representation uses fewer parameters but requires that reference frames be attached to each link following special rules of assignment, which can be cumbersome. Details of the transformation from the D–H to the PoE representation are also provided in Appendix C.

Chapter 5: Velocity Kinematics and Statics

Velocity kinematics refers to the relationship between the joint linear and angular velocities and those of the end-effector frame. Central to velocity kinematics is the **Jacobian** of the forward kinematics. By multiplying the vector of joint-velocity rates by this configuration-dependent matrix, the twist of the end-effector frame can be obtained for any given robot configuration. **Kinematic singularities**, which are configurations in which the end-effector frame loses the ability to move or rotate in one or more directions, correspond to those configurations at which the Jacobian matrix fails to have maximal rank. The **manipulability ellipsoid**, whose shape indicates the ease with which the robot can move in various directions, is also derived from the Jacobian.

Finally, the Jacobian is also central to static force analysis. In static equilibrium settings, the Jacobian is used to determine what forces and torques need to be exerted at the joints in order for the end-effector to apply a desired wrench.

The definition of the Jacobian depends on the representation of the end-effector velocity, and our preferred representation of the end-effector velocity is as a six-dimensional twist. We touch briefly on other representations of the end-effector velocity and their corresponding Jacobians.

Chapter 6: Inverse Kinematics

The **inverse kinematics** problem is to determine the set of joint positions that achieves a desired end-effector configuration. For open-chain robots, the inverse kinematics is in general more involved than the forward kinematics: for a given set of joint positions there usually exists a unique end-effector position and orientation but, for a particular end-effector position and orientation, there may exist multiple solutions to the joint positions, or no solution at all.

In this chapter we first examine a popular class of six-dof open-chain structures whose inverse kinematics admits a closed-form analytic solution. Iterative numerical algorithms are then derived for solving the inverse kinematics of general open chains by taking advantage of the inverse of the Jacobian. If the open-chain robot is **kinematically redundant**, meaning that it has more joints than the dimension of the task space, then we use the pseudoinverse of the Jacobian.

Chapter 7: Kinematics of Closed Chains

While open chains have unique forward kinematics solutions, closed chains often have multiple forward kinematics solutions, and sometimes even multiple solutions for the inverse kinematics as well. Also, because closed chains possess both actuated and passive joints, the kinematic singularity analysis of closed chains presents subtleties not encountered in open chains. In this chapter we study the basic concepts and tools for the kinematic analysis of closed chains. We begin with a detailed case study of mechanisms such as the planar five-bar linkage and the Stewart–Gough platform. These results are then generalized into a systematic methodology for the kinematic analysis of more general closed chains.

Chapter 8: Dynamics of Open Chains

Dynamics is the study of motion taking into account the forces and torques that cause it. In this chapter we study the dynamics of open-chain robots. In analogy to the notions of a robot's forward and inverse kinematics, the **forward dynamics** problem is to determine the resulting joint accelerations for a given set of joint forces and torques. The **inverse dynamics** problem is to determine the input joint torques and forces needed for desired joint accelerations. The dynamic equations relating the forces and torques to the motion of the robot's links are given by a set of second-order ordinary differential equations.

The dynamics for an open-chain robot can be derived using one of two approaches. In the Lagrangian approach, first a set of coordinates – referred to as generalized coordinates in the classical dynamics literature – is chosen to parametrize the configuration space. The sum of the potential and kinetic energies of the robot's links are then expressed in terms of the generalized coordinates and their time derivatives. These are then substituted into the **Euler–Lagrange equations**, which then lead to a set of second-order differential equations for the dynamics, expressed in the chosen coordinates for the configuration space.

The **Newton–Euler** approach builds on the generalization of $f = \mathfrak{m}a$, i.e., the equations governing the acceleration of a rigid body given the wrench acting on it. Given the joint variables and their time derivatives, the Newton–Euler approach to inverse dynamics is: to propagate the link velocities and accelerations outward from the proximal link to the distal link, in order to determine the velocity and acceleration of each link; to use the equations of motion for a rigid body to calculate the wrench (and therefore the joint force or torque) that must be acting on the outermost link; and to proceed along the links back toward the base of the robot, calculating the joint forces or torques needed to create the motion of each link and to support the wrench transmitted to the distal links. Because of the open-chain structure, the dynamics can be formulated recursively.

In this chapter we examine both approaches to deriving a robot's dynamic equations. Recursive algorithms for both the forward and inverse dynamics, as well as analytical formulations of the dynamic equations, are presented.

Chapter 9: Trajectory Generation

What sets a robot apart from an automated machine is that it should be easily reprogrammable for different tasks. Different tasks require different motions, and it would be unreasonable to expect the user to specify the entire time-history of each joint for every task; clearly it would be desirable for the robot's control computer to "fill in the details" from a small set of task input data.

This chapter is concerned with the automatic generation of joint trajectories from this set of task input data. Formally, a trajectory consists of a **path**, which is a purely geometric description of the sequence of configurations achieved by a robot, and a **time scaling**, which specifies the times at which those configurations are reached.

Often the input task data is given in the form of an ordered set of joint values, called control points, together with a corresponding set of control times. On the basis of this data the trajectory generation algorithm produces a trajectory for each joint which satisfies various user-supplied conditions. In this chapter we focus on three cases: (i) point-to-point straight-line trajectories in both joint space and task space; (ii) smooth trajectories passing through a sequence of timed "via points"; and (iii) time-optimal trajectories along specified paths, subject to the robot's dynamics and actuator limits. Finding paths that avoid collisions is the subject of the next chapter on motion planning.

Chapter 10: Motion Planning

This chapter addresses the problem of finding a collision-free motion for a robot through a cluttered workspace, while avoiding joint limits, actuator limits, and other physical constraints imposed on the robot. The **path planning** problem is a subproblem of the general motion planning problem that is concerned with finding a collision-free path between a start and goal configuration, usually without regard to the dynamics, the duration of the motion, or other constraints on the motion or control inputs.

There is no single planner applicable to all motion planning problems. In this chapter we consider three basic approaches: grid-based methods, sampling methods, and methods based on virtual potential fields.

Chapter 11: Robot Control

A robot arm can exhibit a number of different behaviors depending on the task and its environment. It can act as a source of programmed motions for tasks such as moving an object from one place to another, or tracing a trajectory for manufacturing applications. It can act as a source of forces, for example when grinding or polishing a workpiece. In tasks such as writing on a chalkboard, it must control forces in some directions (the force pressing the chalk against the board) and motions in other directions (the motion in the plane of the board).

In certain applications, e.g., haptic displays, we may want the robot to act like a programmable spring, damper, or mass, by controlling its position, velocity, or acceleration in response to forces applied to it.

In each of these cases, it is the job of the robot controller to convert the task specification to forces and torques at the actuators. Control strategies to achieve the behaviors described above are known as **motion** (or **position**) **control**, **force control**, **hybrid motion–force control**, and **impedance control**. Which of these behaviors is appropriate depends on both the task and the environment. For example, a force-control goal makes sense when the end-effector is in contact with something, but not when it is moving in free space. We also have a fundamental constraint imposed by the mechanics, irrespective of the environment: the robot cannot independently control both motions and forces in the same direction. If the robot imposes a motion then the environment determines the force, and vice versa.

Most robots are driven by actuators that apply a force or torque to each joint. Hence, precisely controlling a robot requires an understanding of the relationship between the joint forces and torques and the motion of the robot; this is the domain of dynamics. Even for simple robots, however, the dynamic equations are complex and dependent on a precise knowledge of the mass and inertia of each link, which may not be readily available. Even if it were, the dynamic equations would still not reflect physical phenomena such as friction, elasticity, backlash, and hysteresis.

Most practical control schemes compensate for these uncertainties by using **feedback control**. After examining the performance limits of feedback control without a dynamic model of the robot, we study motion control algorithms, such as **computed torque control**, that combine approximate dynamic modeling with feedback control. The basic lessons learned for robot motion control are then applied to force control, hybrid motion–force control, and impedance control.

Chapter 12: Grasping and Manipulation

The focus of earlier chapters is on characterizing, planning, and controlling the motion of the robot itself. To do useful work, the robot must be capable of manipulating objects in its environment. In this chapter we model the contact between the robot and an object, specifically the constraints on the object motion imposed by a contact and the forces that can be transmitted through a frictional contact. With these models we study the problem of choosing contacts to immobilize an object by **form closure** and **force closure** grasping. We also apply contact modeling to manipulation problems other than grasping, such as pushing an object, carrying an object dynamically, and testing the stability of a structure.

Chapter 13: Wheeled Mobile Robots

The final chapter addresses the kinematics, motion planning, and control of wheeled mobile robots and of wheeled mobile robots equipped with robot arms. A mobile robot can use specially designed **omniwheels** or **mecanum wheels** to achieve omnidirectional motion, including spinning in place or translating in any direction. Many mobile bases, however, such as cars and differential-drive robots, use more typical wheels, which do not slip sideways. These no-slip constraints are fundamentally different from the loop-closure constraints found in closed chains; the latter are **holonomic**, meaning that they are configuration constraints, while the former are **nonholonomic**, meaning that the velocity constraints cannot be integrated to become equivalent configuration constraints.

Because of the different properties of omnidirectional mobile robots versus nonholonomic mobile robots, we consider their kinematic modeling, motion planning, and control separately. In particular, the motion planning and control of nonholonomic mobile robots is more challenging than for omnidirectional mobile robots.

Once we have derived their kinematic models, we show that the **odometry** problem – the estimation of the chassis configuration based on wheel encoder data – can be solved in the same way for both types of mobile robots. Similarly, for mobile manipulators consisting of a wheeled base and a robot arm, we show that feedback control for **mobile manipulation** (controlling the motion of the end-effector using the arm joints and wheels) is the same for both types of mobile robots. The fundamental object in mobile manipulation is the Jacobian mapping joint rates and wheel velocities to end-effector twists.

Each chapter concludes with a summary of important concepts from the chapter, and Appendix A compiles some of the most used equations into a handy reference. Videos supporting the book can be found at the book's website, `http://modernrobotics.org`. Some chapters have associated software, downloadable from the website. The software is meant to be neither maximally robust nor efficient but to be readable and to reinforce the concepts in the book. You are encouraged to read the software, not just use it, to cement your understanding of the material. Each function contains a sample usage in the comments. The software package may grow over time, but the core functions are documented in the chapters themselves.